Managing the Mappings between Domain Ontologies and Database Schemas when Formulating Relational Queries

Kamran Munir
University of the West of England
Centre for Complex Coop. Systems
BS16 1QY, Bristol, UK
+44-1173-283279
Kamran.Munir@cern.ch

Mohammed Odeh
University of the West of England
Centre for Complex Coop. Systems
BS16 1QY, Bristol, UK
+44-1173-283700
Mohammed.Odeh@uwe.ac.uk

Richard McClatchey
University of the West of England
Centre for Complex Coop. Systems
BS16 1QY, Bristol, UK
+44-1173-283176
Richard.McClatchey@cern.ch

ABSTRACT

In recent years, the tremendous increase in the use of medical knowledge-discovery and decision-support applications has often required clinical researchers to write complex database queries. The users of these data analysis systems are normally unaware of the semantic relationships between the concepts stored in a database. In order to provide automated query formulation services, some mechanism for generating queries is required. In this regard, as reported in [1], domain ontologies can be used to formulate relational database queries in order to simplify the data access of the underlying data sources. However, the provision of such a query generation facility requires managing complex mappings between domain ontologies and relational data sources. In this regard, this paper discusses our approach to define mappings between domain ontologies and database schemas to support the ontology assisted relational query formulation process. This approach has been applied to the integrated medical database schema of the EU funded Health-e-Child (HeC) project.

Keywords

Relational Databases, Query Formulation, Domain Ontology, Domain Metadata

1. INTRODUCTION

1.1. The Problem in General

In information management systems, structured query formulation languages are one of the means to retrieve information. Formulated queries allow the selection of data under particular constraints. In contrast to the menu driven (MD) or query by example (QBE) information access methods [2], writing structured queries is a powerful method to access data because it allows end-users to formulate complex database queries and this consequently forces end-users to learn specialised query languages. Thus structured query formulation, with the exception of a few visual query generation approaches, remains noticeably difficult for large classes of end users.

IDEAS09 2009, September 16-18, Cetraro, Calabria [Italy]
Editor: Bipin C. DESAI

Despite the variety of approaches presented so far, three major concerns can be raised when we request information extraction from available data: (1) what type of requests can a specific system handle? (2) how can visual interfaces be provided to generate data requests? and (3) how can the user be assisted in formulating queries in order to retrieve more accurate information? Information technology today has been widely adopted in resolving the first two problems by providing some theoretical and practical solutions using artificial intelligence techniques and graph theories, especially in providing visual tools to generate specific queries. However, little has been achieved in the use of computational techniques to provide users with '*query formulation'* services using '*domain ontologies'*.

For example, a clinical researcher may want to perform a study on patients' '*infections'*. In doing so, an associated domain ontology based system should recognise that '*bacterial'* and '*viral'* are types of infections but '*meningitis'*, '*rat bite fever'*, and '*scarlet fever'* etc. are sub-types of bacterial infection. Here, the system should also identify how the associated data are structured in the underlying database in order to transparently retrieve the resultant dataset. Similarly, for a particular clinical study of '*female'* patients diagnosed with the medical disorder '*double-vision'* and who were using *'anti-depression drugs',* the associated domain ontology based system should recognise that *'double vision'* is a type of clinical test with the possible values *'true'* (to confirm the affliction) or *'false'* (to disprove its existence) and also to check for the medical history of female patients prescribed with *'antidepression drugs'* to retrieve the desired resultant dataset.

1.2. Research Aims

In relation to the above problem, this research aims: (1) to assist end users in formulating relational database queries without requiring a complete knowledge of the information structure and access mechanisms to the underlying data sources; and (2) to enable the developed query formulation methods to : (a) be flexible in terms of accommodating changes in the underlying database schema; and (b) provide access to existing relational database without manipulating or replicating the transactional data. These research aims have been achieved by building an ontology to be the repository for end user queries. Precisely, the emphasis has been put on exploiting the semantic relationships and assertion capabilities of OWL-DL domain ontologies to assist in generating relational database queries. In this regard, the ontology assisted query formulation architectural framework with

several heuristic based ontology to relational query translation algorithms have already been reported in [1] and [3]. This paper reports on further developments in this area and particularly focuses on (1) domain metadata representation from a relational model to an ontology model to enable ontology based query formulation; (2) storage and retrieval of ontology-database mappings to translate ontology statements into relational query statements as per the underlying schema structure; and (3) an ontology assisted query formulation case study in relation to the Health-e-Child [4] project.

1.3. Semantic Query Formulation in Health-e-Child

The Health-e-Child (HeC) project [5] aims to develop an integrated platform for European paediatrics, enabling data integration between spatially distributed clinicians and bringing together information produced in different departments or multiple hospitals. The emphasis of the HeC data integration process is on providing "universality of information". Its cornerstone is the integration of information across biomedical abstractions, whereby all layers of biomedical information can be 'vertically integrated' [6].

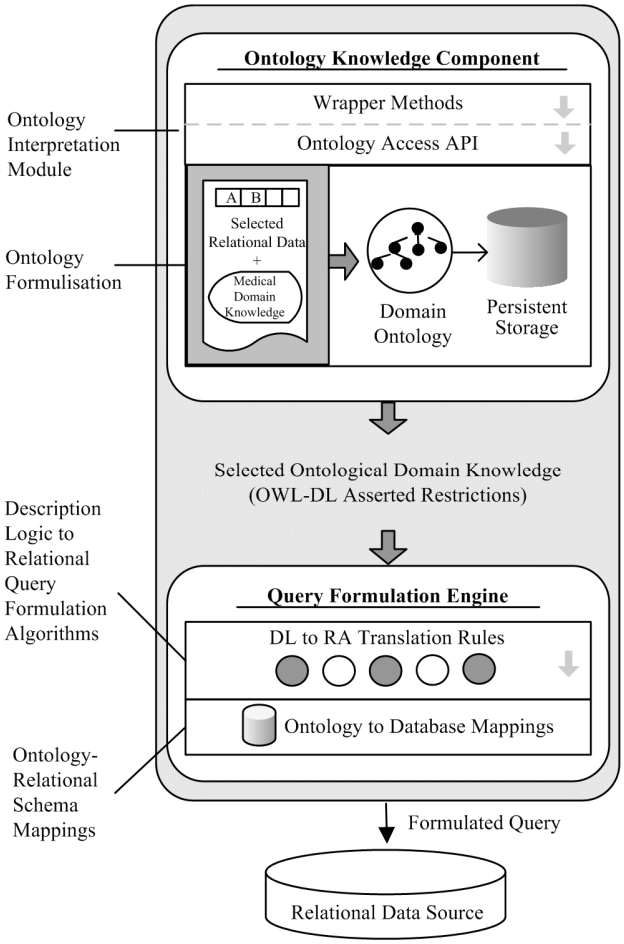

Figure 1: The *OntoQF* architectural framework

The provision of semantic query formulation services in HeC aims at the provision of semantics-driven query formulation services for the clinical researchers and medical knowledge discovery applications to access the HeC Integrated Data Model (IDM) [7]. This task of query formulation has been automated by the successive incremental development of algorithms to test the extent to which this procedure could be effectively automated. In order to test our query formulation system and the developed methodologies, a detailed case study has been implemented (reported in Section 7) using a large subset of the Health-e-Child (HeC) patients' medical data collected during the requirements gathering and data-collection phases of the HeC project. Figure 1 shows the *Ontology Assisted Relational Query Formulation (OntoQF)* architectural framework as reported in [1].

2. RELATED WORK

2.1. Ontology Based Information Retrieval

In recent years, considerable work has been reported that supports ontology based information retrieval. Most of these approaches use RDF [8], [9], [10] and [11] structures which, although yielding schema information, provide insufficient knowledge for database query formulation. These approaches also lack the details of what needs to be included in the ontology from the data sources along with the domain knowledge to drive the process of query formulation. The focus of these approaches (for example [11]) remains towards interactive query generation through nondirected graphs supporting multiple natural languages. Moreover, RDF is too weak to describe resources in sufficient detail since it lacks localised range and domain constraints. In *OntoQF*, OWL-DL is the ontology development language that is used to specify the concepts with related assertions that drive the process of query formulation, since it has explicit support for expressing semantics when compared to RDF and RDFS. Moreover, unlike the approaches in [12], [13], [14], [15] and [16] our system does not store all data from a data-source as part of the ontology, as it may not be practically feasible to store all data as part of a certain domain ontology especially for systems with large amounts of data.

2.2. The *Database-Ontology* Transformation

Currently, there are several theories and tools available that can transform a relational database into an ontology. For example, considerable work has been carried out in [17] and [18] on the transformations between relational databases and ontologies. Most of these transformations are fairly trivial: each table maps to one or more ontology concept(s); each column maps to a datatype property; and each row maps to an instance. In relation to this, the work that has also been carried out in [19] describes the relationship between entities in the entity-relationship (ER) model and DL theory.

In OntoQF, while using domain ontology to formulate relational queries, some of the basic rules to generate domain ontology from relational schema remain the same as reported in [17] and [18]. However, such transformation approaches do not further assist in specifying concept restrictions to generate precise database queries. In our approach, the existing mapping schemes are extended to support query formulation needs by introducing further semantic groupings with respect to cardinality relationships between domain metadata. Moreover, our relational schema to ontology transformation is different in the sense that

transactional data are not transformed and then stored as ontology instances.

2.3. The *Ontology-Database* Transformation/Mappings Approaches

Ontologies allow interaction between data held in different formats and can possibly be used as the basis to guide and validate models of particular domains, for example, conceptual data models [20]. In recent years, some valuable work has been reported which aims to transform ontologies to conceptual data models (expressed, for example, in UML or in ER) in [21], [22], [23] and [24]. However, the ontology-database mappings requirement for query formulation as reported in this paper is rather different than these approaches, because our focus is not to generate conceptual data models from an ontology but to use these mappings to generate domain specific relational database queries. Helpfully in [24] several mapping rules have been proposed that guide the transformation from domain ontology to conceptual schema. One of these mapping rules describes the transformation of ontology properties to entities-attributes in the conceptual data model. In this paper, this rule has been extended to define mappings between an OWL ontology to a data source schema.

3. ONTOLOGICAL REPRESENTATION OF DOMAIN METADATA FOR QUERY FORMULATION

In OntoQF, the process of domain metadata representation in an ontology based on a relational database involves analysing the database schema to determine the database '*domain metadata*' to '*domain ontology*' transformation dependencies. This analysis helps in determining the relational entities to be transformed into ontology concepts. It also helps either to group together or separate the ontological representation of a domain entity and also to determine the relationships between different entities. However, the database-ontology transformation is also dependent on the query formulation requirements.

In general, the term database-ontology (or ontology-database) mapping(s) assumes the existence of both a relational database and an ontology and defines links between them, whereas the database-ontology transformation assumes that only a relational database exists and an ontology is generated by applying database-ontology transformation rules [25]. In OntoQF (as shown in Figure 2), first a domain ontology is created by transforming the database entities which contain the domain metadata and associated semantics. At this stage, the ontology-database mappings (correspondence) entries are also stored. The term *'domain metadata'* refers to those database relations which contain the entity related data or semantics of a domain. The database tables that are generated as a result of creating a relationship between individual entities are called *'transaction tables'* usually denoted by *R(R)*. Such relations are not stored in the domain ontology as classes. This is because we do not aim to migrate or replicate transactional data in the ontology rather the entity related metadata is to be used for querying the transactional data. However, the extent to which domain metadata and semantics are to be represented in a domain ontology can be decreased or increased as per the query formulation requirements.

While creating a domain ontology, the development of a class hierarchy and defining properties of concepts (slots) are closely intertwined. In general, the definitions of the concepts are to be created first in the hierarchy and then the properties of these concepts are described. In this section, we discuss the mapping rules which explain how selected sets of relational entities containing domain metadata are included in the OntoQF ontology. These database-ontology mappings rules have been devised while keeping the followings three important considerations: (1) the result of the database-ontology transformation adequately describes the original database relationships; (2) in some cases, not all constructs in a relational database may require to be transformed in an ontology; and (3) while representing foreign key metadata relations the transformation should maintain the relationships between concepts. The following subsections explain how the database-ontology transformations of domain metadata are performed. At the end of this section we discuss more advanced issues that need to be considered while creating the OntoQF domain ontology.

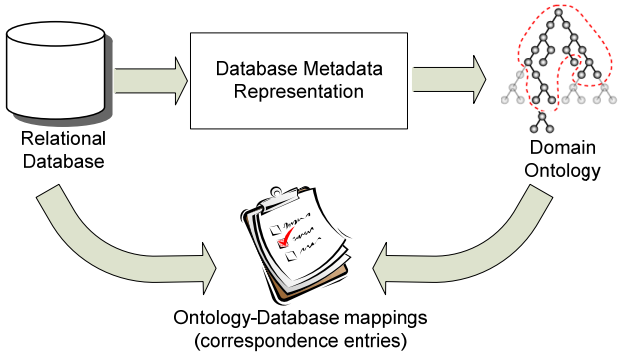

Figure 2: OntoQF domain metadata representation and ontology-database mappings

3.1. Ontological Representation of an Entity Type E containing Domain Metadata

As shown in Figure 3, an entity type E (containing domain metadata) in a relational schema R(E) is represented as a class in the ontological model. Here each distinct column value is stored as a subclass of the entity class. An object property is created which points to the class as property *range*. This rule is only applicable to a standalone entity and does not apply for the representation of domain metadata which is based on any cardinality (1:1 or 1:M etc.) relationship.

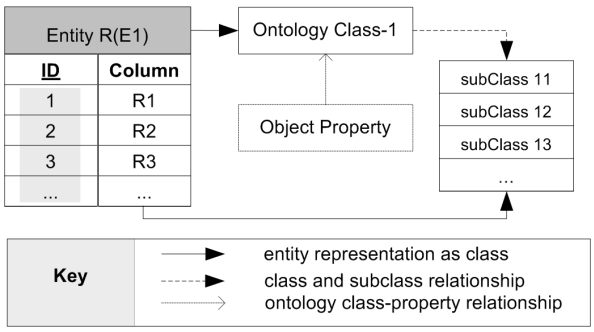

Figure 3: Ontological representation of an entity type E

3.2. Ontological Representation of Domain Metadata with 1:1 Relationships

In such a case, 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 as the primary key of E1 and P2 as the primary key of E2; and R is a one-to-one relationship type between the entity types E1 and E2.

Figure 4 shows entities E1 and E2 in a relational schema R(E). Here, both R(E1) and R(E2) contain domain metadata and there is a 'one to one' relationship from R(E1) to R(E2). In such a situation both R(E1) and R(E2) are required to be represented as ontology classes. For R(E1), an ontology class is created for entity type R(E1) and each entity of the entity type is represented as a subclass of the entity-type class. For R(E2), we have two situations, i.e. (1) if there is a 1:1 g*eneralization (specialization, is-a*) relationship from R(E1) to R(E2), as shown in Figure 4(a), and (2) if there is any other *ID based* 1:1 *relationship (e.g. has-a, part-of etc)* from R(E1) to R(E2), as shown in Figure 4(b).

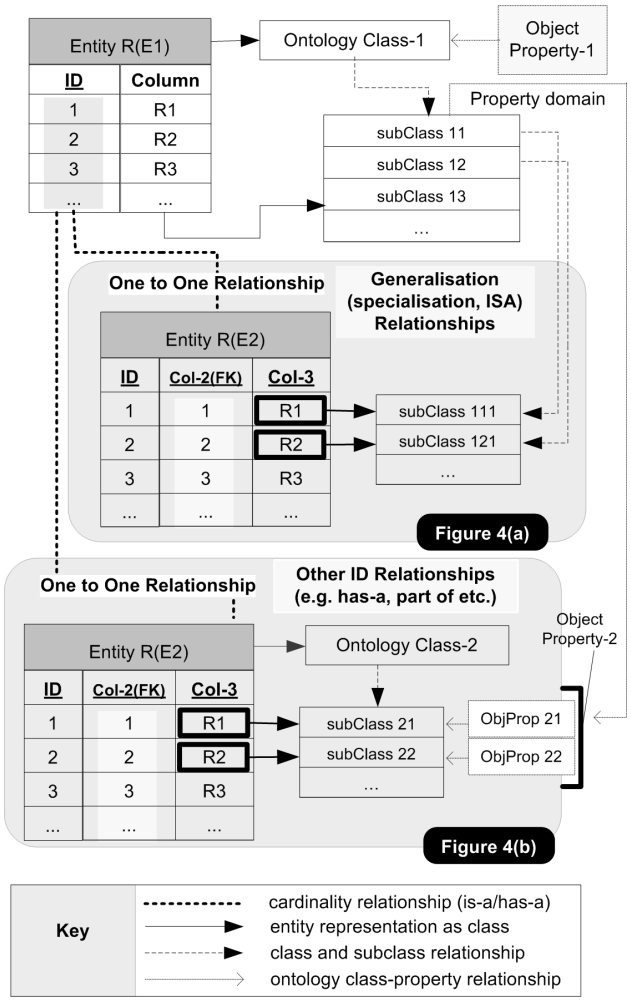

Figure 4: Ontological representation of domain metadata with 1:1 relationships between entity E1 and E2

For an *is-a* relationship, as shown in Figure 4(a), each column value (i.e. col-3) stored as a *foreign key* value is represented as a subclass (i.e. class-111, class-121 etc) of the parent entity class (i.e. class-11, class-12 etc). In this way, all of the R(E2) entities are represented under a generalised class (class-1). The parent class (class-1) is defined as a *range* class for the related object property in order to have each *foreign key* value mapped to a common object property. In addition, similar entities could be further defined under one generalized parent class, if needed. An example of such situation, as in Figure 4(a), is *"Antibiotic drugs"* as parent class and a drug *"Actinomycin"* is an *(is-a)* antibiotic drug.

For all other type of *ID relationships,* as in Figure 4(b), an ontology class (class-2) is created for entity type R(E2) and the column values (i.e. col-3) stored against each *foreign key* are mapped to a subclass (i.e. class-21, class-22 etc) of the entity type class (class-2). In order to link class-1 (primary key values) with class-2 (foreign key values), object properties are used. In order to support the relationship between domain entities within the ontology each subclass of class-2 (i.e. foreign key column values) is linked to an object property (i.e. ObjProp-21, ObjProp-22 etc). All of these object properties are defined by a generalised property (object property-2). Here, the individual properties (objProp21, objprop22 etc) link the subclasses of class-1 (i.e. subclass11, subClass12 etc) to the subclasses of class-2 (i.e. subClass21, subClass22 etc) through property *domain* and *range* relationship. This is how the links are established between a primary key row instances (whose corresponding class is defined as a *domain* class) with a *foreign key* row instance (whose corresponding class is defined as a *range* class). An example of such a situation (as shown in Figure 4(b)) is: a country "*France*" has a *(has-a)* capital "*Paris"*.

3.3. Ontological Representation of Domain Metadata with 1:M Relationships

In Figure 5 both R(E1) and R(E2) contain domain metadata and there is a 'one to many' relationship from R(E1) to R(E2). This situation is almost similar to 1:1 mappings (as detailed in Section 3.2) with the only major difference in the ontological representation of foreign key values for entity R(E2).

In such a case, for an *is-a* relationship, as shown in Figure 5(a), the column values (i.e. col-3) stored against a foreign key are represented as subclasses of the parent entity class (i.e. class-111 and class-112 for class-11, and class-121 for class-12 etc). For all other type of *ID relationships,* as in Figure 5(b), column values (e.g. col-3) stored against one common *foreign key* are represented under a parent class having these values as subclasses (i.e. class 211 and class 212 defined as subclasses for class 21 etc). The parent classes (i.e. class 21, class 22 etc.) are represented by one generalised class (class-2) of R(E2). Here, the object properties are created for each distinct *foreign key* value. All of these object properties are defined by a generalised property (object property-2). Individual properties (objProp21, objprop22 etc) link the subclasses of class-1 (i.e. subclass11, subClass12 etc) to the subclasses of class-2 (i.e. subClass21, subClass22 etc) through property *domain* and *range* relationships. This is how links are established between a primary key row instances (whose corresponding class is defined as *domain* class) with a *foreign key* row instance (whose corresponding classes are defined as *range* classes). An example of such a situation (as shown in Figure 5(b)) is: a country "*France*" has cities "*Paris", "Lyon" etc*.

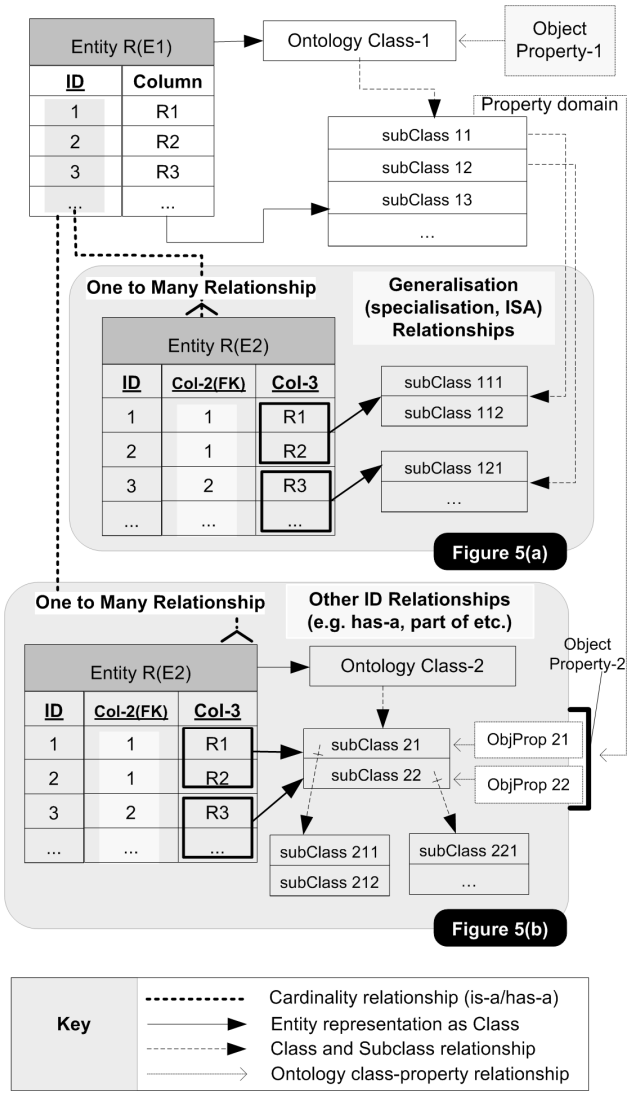

Figure 5: Ontological representation of domain metadata with 1:M relationships between entity E1 and E2

3.4. Ontological Representation of Domain Metadata with M:N Relationships between Entity E1 and E2

In such a case, 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; R is a many-to-many relationship type mapped to a schema relation denoted by R(R) = P1 U P2.

Figure 6 shows the entities E1 and E2 in a relational schema R(E) with 'many to many' relationship between them. In such a situation only the entity types R(E1) and R(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 store transactional data in the ontology rather the entity related domain metadata are used to query such transactional data.

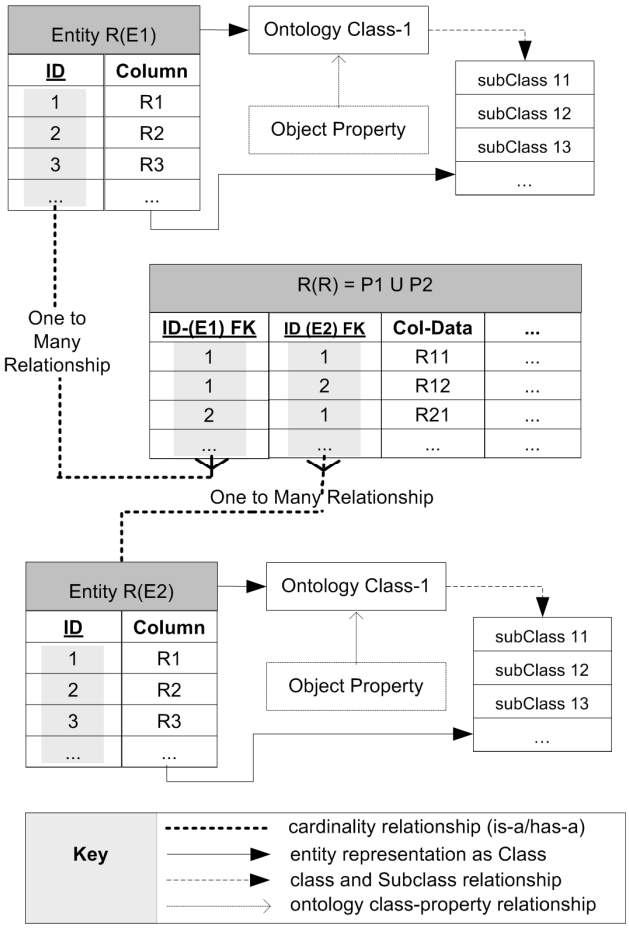

Figure 6: Ontological representation of domain metadata with M:N relationships between entity E1 and E2

3.5. Database to Ontology Mapping for Data Columns

In OWL-DL, property restrictions can be applied to both datatype properties (properties for which the value is a data literal) and object properties (properties for which the value is an individual). Here, 'object properties' link individuals to individuals and 'datatype properties' link individuals to data values.

In OntoQF, table columns whose values are data literals and are not containing domain metadata or semantics are transformed into datatype properties. This is because the table columns which contain domain metadata or semantics are to be defined by class/subclass relationship or by linking them with object properties (as discussed in sections 3.1 – 3.4). Therefore, in order to specify ontology restrictions on data values, for each of the selected table column a datatype property is created which links to the related ontology class as *rdfs:domain*. In such a case, the *rdfs:range* datatype of the 'datatype property' is defined as per the column datatype. Examples of such data columns in the medical domain are: *patient's registration year, patient's disease duration, patient's height, patient's weight* etc. For all of such cases, the *rdfs:domain* class will be the *patient*.

Figure 7 shows an entity E in a relational schema R(E) and the attributes $Col_1, Col_2, ..., Col_n$ belong to entity E. Here, R(E) is represented as an ontology class E and columns (i.e. $Col_1, Col_2, ..., Col_n$) are represented as datatype properties (i.e. $Datatype-Property_1, Datatype-Property_2, ..., Datatype-Property_n$) with class-E defined as *rdfs:domain*. The *rdfs:range* datatype for these ontology properties are defined as per the columns ($Col_1, Col_2, ..., Col_n$) datatype. Such a transformation of data columns can be validated by applying a reverse transformation, i.e. by converting ontology datatype properties to relational database table columns [26]. Here, all datatype properties are parsed in a series. For each parsed datatype property, a database table is located similar to the *rdfs:domain* value for datatype property, and a data column is created with the name of that property.

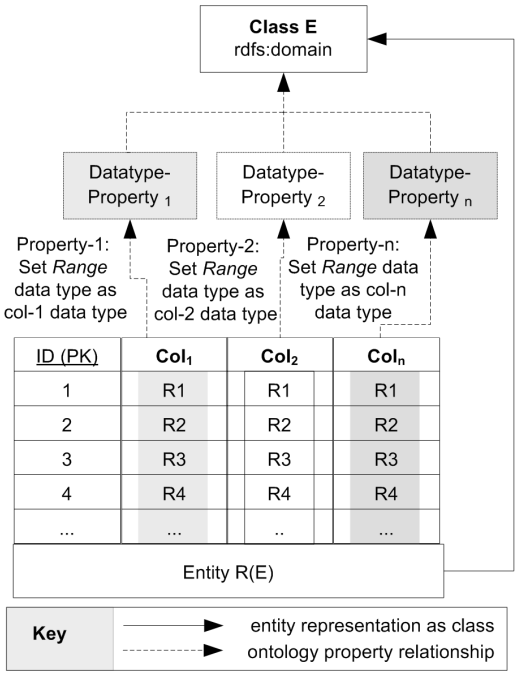

Figure 7: Ontological representation of data columns

4. ONTOLOGY REPRESENTATION OF DOMAIN KNOWLEDGE

Once the basic structural elements of an OntoQF domain ontology have been defined using the above mentioned database-ontology transformation approach, then this ontology can be further enriched with domain knowledge which is expressed in terms of OWL-DL property assertions as concept restrictions that are consistent with the ontology schema. These concept restrictions could be either simple or complex ones, and may involve many conditions. In OntoQF, object properties with quantifier restrictions and datatype properties with comparative restrictions are used for the purpose of generating ontology based queries.

A given DL expression has a set of concepts and a role forming operator. The smallest set propositionally closest to DL is *Attributive Language with Complements (ALC),* where concepts are constructed using *Union, Intersection, allValuesFrom, someValuesFrom and complementOf* written as $\bigcup, \bigcap, \forall, \exists$ and respectively. The '*all'* in *allValuesFrom* is the universal qualifier whereas the '*some'* in *someValuesFrom* is the extensional qualifier. The *someValuesFrom (hasClass)* and *allValuesFrom (toClass)* constructs are applied to classes while specifying classes and restrictions. The cardinality restrictions are referred to as *local* restrictions since they are associated with object properties of a class. That is, restrictions constrain the cardinality of an object property on instances of a class. For example, a *minimum* cardinality of *zero* on an object property states (in the absence of any more specific information) that this property is optional with respect to that class. The cardinality restrictions may also be used to state that certain classes can have no values for a particular property. These cardinality restrictions are used to inform the ontology *Reasoner* [19] of the maximum or minimum number of a class instances that are related to one individual by that object property.

As in OntoQF, the focus remains on using a domain ontology to formulate database queries by translating ontology statements into relational query statements; therefore, cardinality constraints are not used to perform query formulation tasks. This is mainly because these restrictions, when used with object properties, constrain the cardinality of a class property. This is because class instances contain the transactional data of a domain, whereas in OntoQF we do not migrate or replicate transactional data into the ontology. However, the comparative restrictions *minimum, maximum* and *exactly* can be used with the datatype properties to qualify domain knowledge statements. Here, the restrictions constrain the value of a datatype property in relation to data values in a table column. In such a case, *minimum* with a datatype property states (in the absence of any more specific information) that the selected class should not have any value less than the specified value. Moreover, all of these *minimum*, *maximum* and *exactly* restrictions can also be used with a *negation (not)* operator.

5. TRANSLATION OF ONTOLOGY STATEMENTS INTO RELATIONAL QUERY STATEMENTS

As detailed in Sections 3 and 4, the OntoQF domain ontology stores the domain metadata or semantics, and the domain knowledge is expressed in terms of ontology statements as concept restrictions. These concept restrictions can include individual or multiple OWL description logic constructs as well as conditions. In order to translate these ontology statements into relational query statements, each DL construct needs to be processed and translated into the corresponding relational query constructs. The following are the syntax rules for DL constructs [19].

C, D → A | (atomic concept)
I (Instance)
¬ C | (negation)
C ∩ D | (intersection)
C U D | (union)
∀P.C | (universal restriction)
∃P.C | (existential quantification)
P_D . Min/Max/Exactly {value} | (P_D → datatype property)

Table 1: Translating Ontology OWL-DL Statements into Relational Query (RQ) Statements

OWL-DL Formulae	FOL	* Translation in Relational Query Statements
Existential Quantification (some P C) i.e. $\exists$ P.C	$\exists$ y. P(x, y) $\wedge$ C(y) Here y: C $\rightarrow$ C(y) and (x, y) : P $\rightarrow$ P(x, y)	For a DL concept $D_C = \exists$ P.C the set A is called the domain of p(x). Therefore $D_C = (\exists x \in A, P(x))$, i.e. for any existence of x in A such that *P*(x) is true. The equivalent RA expression i.e. $\sigma_{P=C}(R)$ will select values from a relation/view where the corresponding record satisfies this condition. In this case the ontology-database mapping information is to be used to retrieve the target table.column information.
Universal Restriction (all P C) i.e. $\forall$ P.C	$\forall$ y. P(x,y) $\rightarrow$ C(y)	For a DL concept $D_C = \forall$ P.C, the set A is called the domain of p(x). As per DeMorgan's theorem $\forall$ x. p(x) = $\exists$ x. p(x) and by applying negation ($\forall$ x. p(x))= ($\exists$ x. p(x)) is equal to $\forall$ x. p(x) = ($\exists$ x. p(x)) Thus, in order to show that $\forall$ x. p(x) is true, it is equivalent to show that $\exists$ x. p(x) is false. Therefore, in order to find x for which the p(x) is false, the equivalent RA expression will first select values from a relation where the corresponding record satisfies this condition and then taking an inverse *(complementOf)* of the result set. Therefore, for $\forall$ P.C, $Q_R = \sigma_{domain-col \quad IN(\pi_{domain-col}(\sigma_{P<>C}(R)))}(R)$
Operations (a) Intersection i.e. $C_1 \cap \ldots \cap C_n$ (b) Union i.e. $C_1 \cup \ldots \cup C_n$ (c) Negation i.e. C	(a) $C_1(x) \wedge \ldots \wedge C_n(x)$ (b) $C_1(x) \vee \ldots \vee C_n(x)$ (c) C(x)	The *Intersection, Untion* and *Negation* operators in DL (FOL) has same interpretation as *AND* ($\wedge$), *OR* ($\vee$) and *complementOf* ( ) operators respectfully, i.e. (a) If $C_1(x)$ and $C_2(x)$ are true, then $C_1(x) \wedge C_2(x)$ is true; otherwise $C_1(x) \wedge C_2(x)$ is false. (b) If $C_1(x)$ and $C_2(x)$ are false, then $C_1(x) \vee C_2(x)$ is false; otherwise $C_1(x) \vee C_2(x)$ is true. (c) If C(x) is true, then C(x) is false; and if C(x) is false, then C(x) is true.
Object Property with data values (a)Min {value} (b)Max {value} (c)Exactly{value}	(a)$P_D \geq$ {value} (b)$P_D \leq$ {value} (c)P_D = {value}	If C $\in \zeta$	(P_D (*exactly|min|max)* (value)) then (a) $Q_R = \sigma_{P_D \geq value}(R)$, (b) $Q_R = \sigma_{P_D \leq value}(R)$ and (c) $Q_R = \sigma_{P_D = value}(R)$ Here the RA expression is required to select values from a relation/view where the corresponding record satisfies P_D = *value* condition.
Table Key: ***C** is the ontology concept/class and **R** is a database relation/view or multiple database relations with Join condition. The ontology-database schema mapping information (detailed in section 7) is used to replace R with SQL' FROM' clause.* ***P/P$_D$** is an ontology object-property/datatype-property, which relates x to y or to a data value. While translating DL to RA, the P/P_D is replaced with the database schema information (discussed in section 7).* ***Domain-col** is a manually selected domain column of relation R for ontology property P (discussed in section 7).* *A **Predicate function** P: X→ {true, false} is called a predicate on X. When P is a predicate on X, we say P is a property of X.* ***Note: -** For theoretical background and further details on DL to RA translation algorithms readers are referred to [1] reported earlier.		

In the relational database paradigm, a logical data model may be accessed through SQL which is based on the Relational Algebra (RA), whereas OWL-DL is based on Description Logic [27]. Therefore, we base our translations on Description Logic and Relational Algebra, to work with any relational database that implements the SQL standard. In this regard, Table 1 outlines DL to RA translation rules for individual constructs. Due to space limitations, Table 1 presents only the core DL to RA translations; for details, please refer to [1].

While applying DL to RA translation on DL's *existential* quantification or *universal* restriction when specified with an ontology class C, further checks need to be carried out to see if the class C contains subclasses. This is because a database needs to be searched for all individuals which can be defined as instances of class C. In OntoQF, the database-ontology metadata transformation scheme (as detailed in section 3.2 & 3.3) enables such operation by storing the related entity elements *(with "is-a" relationship)* as subclasses. Using this approach, if an ontology class C contains subclasses; the query condition on a class C is to be updated as per the following two situations (a) and (b).

(a) If C contains only one subclass C_1, then class C is to be replaced with C_1

i.e. For $\exists$ P.C i.e. $\sigma_{P=C}(R)$ it will be $\sigma_{P=C_1}(R)$

and for $\forall$ P.C i.e. $\sigma_{domain-col \quad IN(\pi_{domain-col}(\sigma_{P \neq C}(R)}(R)$

it will be $\sigma_{domain-col \quad IN(\pi_{domain-col}(\sigma_{P \neq C_1}(R)}(R)$

(b) If C contains more than one subclasses i.e. $C_1, C_2, \ldots, C_n$ then C is to be replaced with all of the subclasses of C using the *union* operator i.e. $C_1 \cup C_2 \cup \ldots \cup C_n$

i.e. For $\exists$ P.C i.e. $\sigma_{P=C}(R)$

it will be $\sigma_{P=C1 \vee P=C2 \vee .... \vee P=Cn}(R)$

and for $\forall$ P.C i.e.

$\sigma_{domain-col \quad IN(\pi_{domain-col}(\sigma_{P \neq C}(R))}(R)$, it will be

$\sigma_{domain-col \quad IN(\pi_{domain-col}(\sigma_{P \neq C1 \vee P \neq C2 \vee .... \vee P \neq Cn}(R))}(R)$

Here, both (a) and (b) are to be applied recursively, until the last child node of class C is retrieved.

6. Mappings from an Ontology Model to a Relational Model

Once the ontology statement constructs are translated into corresponding relational query statement constructs (as outlined in Section 5), further processing is required to produce a final executable relational query. This includes (1) replacement of the ontology property links (i.e. P/ P_D) as per the underlying database schema structure; and (2) integration of a SQL *From* clause including table *join* conditions (when required). In OntoQF, ontology-database mappings are used to achieve these tasks. The ontology-database mappings are expressed as a set of correspondences that relate the vocabulary of an ontology model (concept, property etc) with a relational model (table(s), column etc). To this end, two database tables containing ontology-database *mappings* information are created during the ontology processing phase. These mapping tables store information about *ontology property links, database name, table name, column name, primary* and *foreign key(s) (i.e. table join information).* Once the query formulation engine translates the OWL description logic constructs into respective relational constructs, this ontology-database mapping information is used to generate a final query. In order to access the ontology-database mapping definitions a *'mapping-access'* scheme is implemented within the query formulation engine (as shown in Figure 1). The ontology-database mappings are stored within the existing ontology server using the same namespace being used to store corresponding ontology model. In this way an ontology file, when processed and stored in the ontology server, has its database mapping information stored alongside it. In the ontology server, the following two database tables (1) *Mappings_Information* and (2) *Mappings_Relationship* (having 1:M relationship) have been implemented to store the ontology-database mappings:

Table-1: *'Mappings_Information'*

Column-Info	*Constraint*
'ID' *(primary key)*	*(Not Null/ Unique)*
'Onto_Property' *(object or datatype)*	*(Not Null/ Unique)*
'Col_Name'	*(Not Null)*
'Table_Name'	*(Not Null)*

Table-2*: 'Mappings_Relationship'*

Column-Info	*Constraint*
'ID' *(primary key)*	*(Not Null/ Unique)*
'Relationship_Table' *(primary key)*	*(Not Null/ Unique)*
'Key_Table_Join'	*(Not Null)*

(Foreign Key ID reference Mappings_Information.ID)

The *'ontology-property'* to *'database-schema'* conversions are straightforward for the queries which only access a single attribute from a database entity/relation. However, some complex processing is required when multiple attributes are to be accessed with different associated constraints in a single query, or when a database query requires a *Join* operation. In this regard, the *mappings_relationship* table provides the *Join* condition with respect to a given ontology statements.

7. Experimental Evaluation: A Case Study from Health-e-Child Project

This approach has been applied to a large subset of the integrated Health-e-Child (HeC) patients' database schema along with the implementation of a prototype system to perform query formulation tasks. The prototype system has been presented to the HeC consortium and validated by domain experts who have confirmed its potential functionality. Due to space limitations, we present a subset of the HeC case study and discuss associated results.

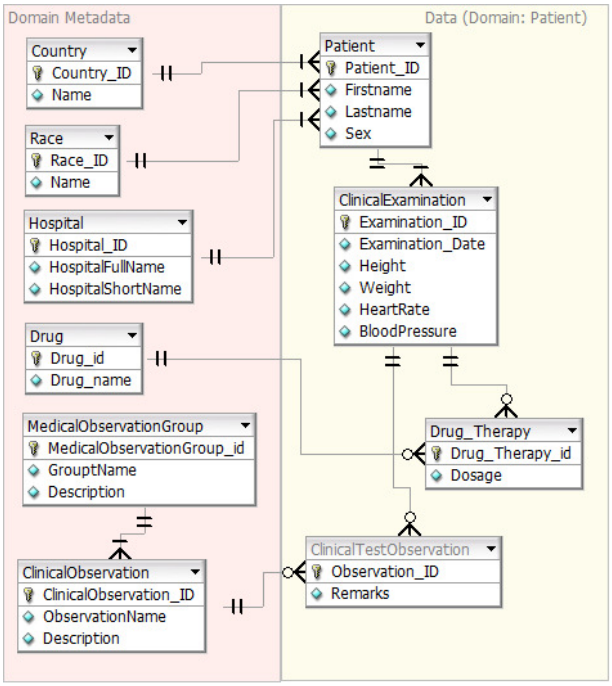

Figure 8: A subset of HeC patients' database showing domain metadata and data tables

This section explains how the HeC domain metadata, used to drive the process of generating queries, is represented in the OntoQF domain ontology. In this regard, Figure 8 shows an example subset of the HeC relational data model. This subset of the data model is separated into two parts: (1) the database tables containing domain metadata; and (2) the database tables which contain transactional data. Here, domain metadata are to be represented into the domain ontology to assist in both: (a) domain knowledge representation; and (b) to generate relational database query statements. As in OntoQF, transactional data are not stored in the ontology rather the domain metadata is used for querying the transactional data. The tables containing HeC domain metadata e.g. *Country, Race, Hospital, Drug, MedicalObservationGroup* and *ClinicalObservation* are transformed into ontology schema (as shown in Figure 9) as per

the database-ontology transformation approach (detailed in section 3). Moreover, due to query formulation requirements the *Gender* column (represented as *Sex*) in the *patient* table is also selected as domain metadata and represented as an ontology concept '*Gender'* having subclasses *'Male'* and '*Female'*. Furthermore, as shown in Figure 9, *'Domain:Patient'* implies that the queries are to be formulated with respect to patient entity.

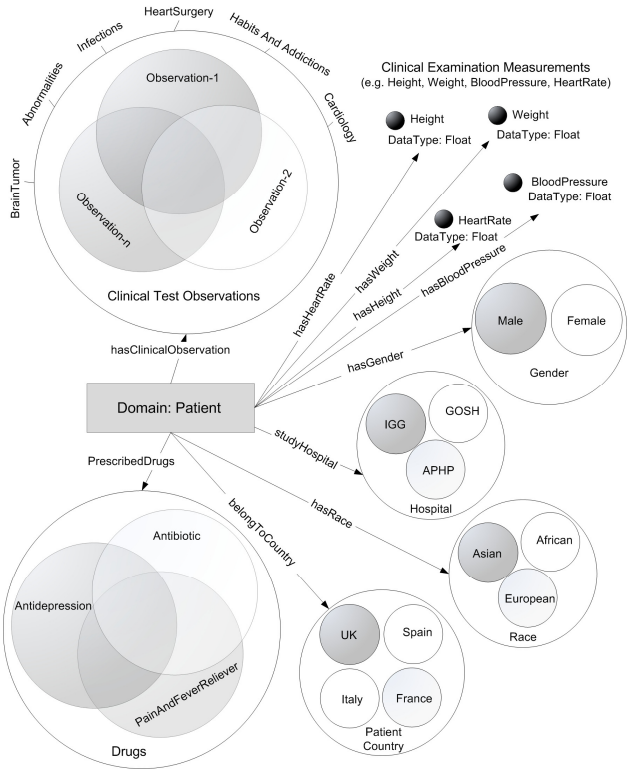

Figure 9: An OntoQF Domain Ontology view of the example Domain Metadata ***(created using the database-ontology mapping scheme to support domain knowledge representation and query formulation)***

Once the OntoQF ontology is formulated, it is further enriched by defining clinical case studies. These clinical case studies were collected during the requirements gathering and data collection phases of the HeC project. In this paper, we have included the following two example case studies. OntoQF is able to automatically generate relational database queries for such (but not limited to) example case studies using the associated OntoQF domain ontology and ontology-database mappings.

Clinical Study 1:

Patients prescribed with only Paracetamol and do Exercise with no other medical problems.

Processed Ontology Statement:

Class Clinical Study-1 is a sub-class of union of {restriction on property :PrescribedDrugs all values from Class :Paracetamol restriction on property :hasClinicalObservation all values from Class :Exercise} is a sub-class of Class :Patients

Generated Relational Query Statement:

SELECT *<<omitted to save space>>*
FROM *patient p, clinicalexamination ce, drug d, drug_therapy dt, clinicaltestobservation cto, clinicalobservation co*
WHERE
p.patient_id = ce.patient_id and
ce.examination_id = cto.examination_id and
ce.examination_id = dt.examination_id and
dt.drug_id = d.drug_id
and (p.patient_id NOT in
(SELECT *p.patient_id* ***FROM*** *patient p, clinicalexamination ce,drug_therapy dt,race r,drug d*
WHERE *p.patient_id = ce.patient_id and ce.examination_id = dt.examination_id and dt.drug_id = d.drug_id and d.Drug_name != "Paracetamol")*
or p.patient_id NOT in
(SELECT *p.patient_id* ***FROM*** *patient p,clinicalexamination ce,clinicaltestobservation cto*
where *p.patient_id = ce.patient_id and co.examination_id = cto.examination_id and* cto.observation_id = co.observation_id *and co.observationname != "Exercise"****));***

Table: Mappings_Information

ID	ONTO_PROPERTY	COL_NAME	TABLE_NAME
1	belongToCountry	Name	Country c
2	hasCO	*ObsName	*CO co
3	hasGender	Sex	Patient p
4	hasrace	Name	Race r
5	prescribedDrugs	Drug_Name	Drug d
7	hasBloodPressure	BloodPressure	*CE ce
8	hasHeartRate	HeartRate	*CE ce
9	hasHeight	Height	*CE ce
10	hasWeight	Weight	*CE ce

Table: Mappings_Relationship

ID	RELATIONSHIP	KEY_JOIN
1	Patient p	c.Country_ID = p.Country_ID
2	Patient p	p.Patient_ID = ce.Patient_ID
2	*CE ce	ce.*E_ID= cto.*E_ID
2	*CTO cto	cto.*O_ID = *co.O_ID
4	Patient p	r.Race_ID = p.Race_ID
5	Patient p	p.Patient_ID = ce.Patient_ID
5	*CE ce	ce.*E_ID = dt.*E_ID
5	Drug_Therapy dt	dt.Drug_id = d.Drug_id
7	Patient p	ce.Patient_ID = p.Patient_ID
…	…	…

***Key**

CE	= ClinicalExamination
CTO	= ClinicalTestObservation
CO	= ClinicalObservation
ObsName	= ObservationName
E	= Examination
O	= Observation

Figure 10: Ontology-Database mappings for the subset of HeC patients' database as per the mapping scheme detailed in Section 7

Clinical Study 2:

Male patients' heart rate reported ≥ 70 and blood pressure ≥ 120 and weight ≤ 75 and never used any physical aids or devices.

Processed Ontology statement:

Class Clinical Study-2 is a sub-class of intersection of { restriction on property :hasGender some values from Class :Male restriction on property :hasClinicalObservation some values from Class :Smoking restriction on property :hasHeartRate min 70 restriction on property :hasBloodPressure min 120 restriction on property :hasWeight max 75 complement of { restriction on property :hasClinicalObservation some values from Class :AidsAndDevices} is a sub-class of Class :Patients

Generated DB Query:

SELECT *<<omitted to save space>>*
FROM *patient p, clinicalexamination ce, clinicaltestobservation cto, clinicalobservation co*
WHERE *p.patient_id = ce.patient_id and ce.examination_id = cto.examination_id and ce.patient_id = p.patient_id and* cto.observation_id = co.observation_id *(patient.sex="Male" and ce.heartrate <= 70 and ce.bloodpressure <= 120 and ce.weight >= 75 and not (co.observationname in ("LongArm" , "WheelChair" , "WalkingStick" , "BathSeat")));*

8. DISCUSSION

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. However, in OntoQF we deal with the *'Select'* query operations exclusively. Therefore, the OntoQF domain ontology does not require the definition of general schema *datatype ranges*, *cardinality restrictions* or *specific constraints (expressed in the database schema)* as axioms. Moreover, while defining a class hierarchy through database-ontology transformation, it is not always required that the ontology should contain all the possible information about the domain.

In some cases, an ontology statement to relational query translation can result in generating *unsafe* relational queries. Here the term '*safe*' implies that the correct answer of the query is contained within (or can be obtained directly from) the associated database tables. For example, multiple *someValuesFrom()* restrictions are used in an ontology restriction, with an *intersection* operation within each restriction and a *similar* property defines each concept. In such a case, the semantics of an *intersection* class means that the resultant class is described as a subclass of interesting class 'A' and a subclass of interesting class 'B' for a *similar* property. However, if the underlying database implements *normalization* i.e. *"a single database record cannot hold more than one value for a table column"* then the generated query from such ontological definition may produce either no or incorrect results. Similarly, another example of such *unsafe* ontology statement is "use of a *complementOf* construct, with an *intersection* operation within each *hasValue* property restriction, i.e. *complementOf (a, b,…,n)* with *intersection* operation". Such *unsafe* ontology definitions can be avoided by applying the following rule: "for a given ontological domain knowledge, if the restriction includes the *'conjunction'* operation then the translation is only *safe* if the *conjunction* is *safe*".

9. CONCLUSIONS

The central aim of this work is to assist clinical researchers in formulating relational database queries using an OWL-DL based domain ontology. The task of query formulation is investigated by the successive incremental development of DL to RA translation algorithms. One of the major requirements of an ontology assisted query formulation system is the formulation of a domain ontology which includes definition of domain metadata, relationships and knowledge of the ontology. In this regard, an ontology modelling approach has been identified to transform domain metadata and relationships into the ontology schema to assist in the query formulation process. Once the basic structural elements of the domain ontology are defined they are further enriched with domain knowledge. Moreover, in order to generate relational query statements as per the underlying database schema structure, ontology-database mappings are expressed as a set of correspondences that relate the vocabulary of a relational model (table/relation, column etc) with the ontology model (concept, property etc). This ontology assisted query formulation approach has been applied to a large subset of the integrated HeC patients' database schema along with the implementation of a prototype system to perform query formulation.

One of the key merits of this approach is that no interpretation of stored transactional data needs to be carried out or even transformed and stored as ontology instances. This is clearly beneficial since the interpretation of data in existing medical data source(s) may cause serious scalability issues with existing legacy clinical applications. Unlike most of the existing database-ontology transformation approaches, the database-ontology mappings presented in this research provide literal representation of the domain-metadata in an ontology. Thus, the ontology generated by following such mappings can play a significant role in specifying concept restrictions and in generating relational database queries. Also, this approach does not require users to be familiar with the overall contents of the ontology to generate respective queries. As a consequence, this can provide significant support to clinical researchers while performing their studies, especially if they do not fully understand the underlying system; and thus navigating in a large ontology to select appropriate terms can be significantly eased rather being a key obstacle in achieving the associated study goals. Finally, we anticipate that the provision of ontology assisted query formulation will provide sufficient theoretical foundation to instantiate this approach in further application domains.

10. ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the *European Union* in funding this work and the valuable assistance of all partners in the *Health-e-Child* project, with special thanks to colleagues working for *University of the West of England (UWE)* at *Center for European Organization for Nuclear Research (CERN)*, Geneva and colleagues at *UWE*.

11. REFERENCES

[1] K. Munir, M. Odeh and R. McClatchey, Ontology Assisted Query Reformulation Using the Semantic and Assertion Capabilities of OWL-DL Ontologies, *Twelfth International Database Engineering & Applications Symposium (IDEAS)* 2008.

[2] M. Zloof, Query-by-example: the invocation and definition of tables and forms, *VLDB: Proceedings of the 1st International Conference on Very Large Data Bases* pp. 1-24, 1975.

[3] K. Munir, M. Odeh, P. Bloodsworth and R. McClatchey, Using Assertion Capabilities of an OWL-Based Ontology for Query Formulation, *3rd International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA)* 2008.

[4] Health-e-Child, The Information Societies Technology Project: Health-e-Child, EU Contract IST-2004-027749, 2004.

[5] J. Freund, Health-e-Child: An Integrated Biomedical Platform for Grid-Based Pediatric Applications, vol. Studies in Health Te, pp. 259-270, 2006.

[6] A. Anjum, P. Bloodsworth, A. Branson, T. Hauer, R. McClatchey, K. Munir, D. Rogulin and J. Shamdasani, The Requirements for Ontologies in Medical Data Integration: A Case Study, *Eleventh International Database Engineering & Applications Symposium (IDEAS)* vol. 6, pp. 308-314, 2007.

[7] A. Branson, T. Hauer, R. Mcclatchey, D. Rogulin and J. Shamdasani, A Data Model for Integrating Heterogeneous Medical Data in the Health-e-Child Project, *Accepted in HealthGrid'08 Conference* 2008.

[8] N.W. Paton, R. Stevens, P. Baker, C.A. Goble, S. Bechhofer and A. Brass, Query Processing in the TAMBIS Bioinformatics Source Integration System, *Proceedings of the IEEE International Conference on Scientific and Statistical Databases (SSDBM)* pp. 138-147, 1999.

[9] E. Mena, A. Illarramendi, V. Kashyap and A. Sheth, OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies, *Journal on Distributed and Parallel Databases* vol. 8, no. 2, pp. 223-271, 2000.

[10] D. Baer, P. Groenewoud, E. Kapetanios and S. Keuser, A Semantics Based Interactive Query Formulation Technique, *User Interfaces to Data Intensive Systems: Second International Workshop on User Interfaces to Data Intensive Systems* pp. 43-49, 2001.

[11] E. Kapetanios, D. Baer, B. Glaus and P. Groenewoud, Data Querying and Analysis through Integration of Intentional and Extensional Semantics, *16th International Conference on Scientific and Statistical Database Management (SSDBM)* pp. 353-356, 2004.

[12] C.B. Necib and J.-C. Freytag, Query Processing using Ontologies, *CAiSE* pp. 167-186, 2005.

[13] A.L. Rector, S. Bechhofer, C.A. Goble, I. Horrocks, W.A. Nowlan and W.D. Solomon, The GRAIL Concept Modelling Language for Medical Terminology, *Artificial Intelligence in Medicine* vol. 9, pp. 139-171, 1997.

[14] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi and R. Rosati, Ontology-based Database Access, *Proc. of the 15th Italian Conf. on Database Systems (SEBD)* pp. 324-331, 2007.

[15] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi and R. Rosati, Linking Data to Ontologies: The Description Logic DL-LiteA, *In Proc. of the 2nd Workshop on OWL: Experiences and Directions (OWLED)* 2006.

[16] Y. Arens, C.A. Knoblock and W.-M. Shen, Query Reformulation for Dynamic Information Integration, *Journal of Intelligent Information Systems - Special Issue on Intelligent Information Integration* vol. 6, no. 2, pp. 99-130, 1996.

[17] Z. Xu, S. Zhang and Y. Dong, Mapping between Relational Database Schema and OWL Ontology for Deep Annotation, *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on Web Intelligence* pp. 248-552, 2006.

[18] J. Barrasa, O. Corcho, G. Shen and A. Gomez-Perez, R2O, an Extensible and Semantically Based Database-to-ontology Mapping Language, *2nd Workshop on Semantic Web and Databases (SWDB)* 2004.

[19] A. Borgida, M. Lenzerini and R. Rosati, Description Logics for Databases, *The description logic handbook: theory, implementation, and applications* pp. 462-484, 2003.

[20] D. Caragea, J. Pathak, J. Bao, A. Silvescu, C. Andorf, D. Dobbs and V. Honavar, Information Integration from Semantically Heterogeneous Biological Data Sources, *DEXA Workshops: Proceedings of the 3rd International Workshop on Biological Data Management* pp. 580-584, 2005.

[21] E. Vysniauskas and L. Nemuraite, Transforming Ontology Representation from OWL to Relational Database, *Information Technology and Control* vol. 35, no. 3A, 2006.

[22] J. Trinkunas and O. Vasilecas, A Graph Oriented Model for Ontology Transformation into Conceptual Data Model, *Journal of Information Technology and Control* vol. 36, no. 1A, 2007.

[23] P. Mitra, G. Wiederhold and M. Kersten, A Graph Oriented Model for Articulation of Ontology Interdependencies, *Proc. Extending Database Technologies* vol. 1777, pp. 86-100, 2000.

[24] H. El-Ghalayini, M. Odeh, R. McClatchey and T. Solomonides, Reverse Engineering Domain Ontologies to Conceptual Data Models, *Proceedings of the 23rd IASTED International Conference on Databases and Applications* pp. 222-227, 2005.

[25] I. Astrova, N. Korda and A. Kalja, Rule-Based Transformation of SQL Relational Databases to OWL Ontologies, *2nd International Conference on Metadata & Semantic Research* 2007.

[26] E. Vysniauskas and L. Nemuraite, Transforming Ontology Representation from OWL to Relational Database, *Information Technology and Control* vol. 35, no. 3A, pp. 333-343, 2006.

[27] F. Baader, I. Horrocks and U. Sattler, Description Logics as Ontology Languages for the Semantic Web, *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg Siekmann, in Lecture Notes in Artificial Intelligence* vol. 2605, pp. 228-248, 2005.