Overcoming database heterogeneity to facilitate social networks: the Colombian displaced population as a case study

Juan F. Sequeda* Alexander Garcia-Castro[§] Oscar Corcho[ϕ]
Syed Hamid Tirmizi* Daniel P. Miranker*

ABSTRACT
In this paper we describe a two-step approach for the publication of data about displaced people in Colombia, whose lack of homogeneity represents a major barrier for the application of adequate policies. This data is available in heterogeneous data sources, mainly relational, and is not connected to social networking sites. Our approach consists in a first step where ontologies are automatically derived from existing relational databases, exploiting the semantics underlying the SQL-DDL schema description, and a second step where these ontologies are aligned with existing ontologies (FOAF in our example), facilitating a better integration of data coming from multiple sources.

Categories and Subject Descriptors
H.2.4 [**Systems**]: Relational Databases

General Terms
Management, Experimentation, Languages

Keywords
Semantic Web, Data Integration, FOAF, Displaced Population

1. INTRODUCTION
Social networks (e.g., Facebook, LinkedIn, etc.) facilitate people's interaction by electronic means; the dynamics within social networks is simple, the sites offer a vast variety of tools so members of the networks can install them and generate or import diverse personal and non-personal content. Users find other users based on the published content and installed tools. For instance users can be notified about other users with similar interests within their geographical vicinity.

Central to the notion of social networks it is precisely "the things they create and do" as well as "word of mouth being supported by electronic and digital means"; in summary, user generated content being distributed across networks. This implies that these systems heavily rely on the community actively participating in the generation of content.

*Department of Computer Sciences, University of Texas at Austin, USA.
Email: {jsequeda, hamid, miranker} @ cs.utexas.edu
[§]Department of Computational Linguistics, University of Bremen, Germany.
Email: cagarcia@uni-bremen.de
[ϕ] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain.
Email: ocorcho@fi.upm.es

Copyright is held by the author/owner(s).
WWW 2009, April 20-24, 2009, Madrid, Spain.

Social networks conceived in this manner tend to be human-centric and mediated by high tech; the load of interaction is facilitated by the applications and it relays on the engagement of the community. The community shares information; the interaction is thus mediated by the degree of interest a particular piece of information may arise in others with similar interests or problems. There are scenarios for which the interaction amongst the community also involves third party actors –external to the community. For instance, the displaced population in Colombia, estimates, though widely different, places the total displaced population between 1'506.869 to 3'100.000, corresponding to 3.4 or 7.1 percent of the country's population. This population is highly vulnerable, fragmented, without economical means; as a community, also involves and requires Non Governmental Organizations (NGOs) and government agencies to be considered as part of the network. Enabling interaction for this scenario requires both, **facilitating interoperability across heterogeneous data sources** and **delivering these integrative views** by means of a mixture of low and high tech, which can be achieved by means of many of the **social networking sites** currently active in the world.

The Colombian case is a challenging one since it is a low intensity conflict; in economic terms the policies toward displaced population are demand driven rather than supply driven. For high intensity conflicts displacement is of a massive scale, usually requiring the State or the international community to set up special camps for providing assistance and safe refuge. The Colombian government has setup aid packages for people that have been displaced due to violence. However, due to the characteristics mentioned before, in order to have access to government aid the displaced individuals must approach the government agency in charge and be registered (Sistema Único de Registro, Unique Registry System, SUR). After being registered the local representative of this central agency evaluates the information and within 15 days it is determined if the household is recognized as displaced and hence it has access to the government aid. A recent evaluation of the SUR system revealed that the exclusion from government aid (through SUR) greatly depends on the displaced households actions and characteristics rather than institutional factors. Furthermore, the SUR does not provide any facility that allows this government agency to exchange information with any of the existing NGOs currently working with displaced population. The dependency that is created between victims, NGOs and government agencies is one of the greatest challenges low intensity conflicts have. In general, these conflicts force institutions to locate and track the displaced population not only for delivering goods and services, but also to

define the policies that will allow NGOs and government agencies to define action policies to deal with the problem. In order for demand driven aid systems to work properly information should flow across displaced population, NGOs and government agencies. Victims should be able to access information about the aid available for them, by the same token NGOs and government agencies should be able to gather the information they need, verify and cross reference it in an efficient and timely manner.

This is a two fold problem affecting both, the community and the policy makers; on the one hand there is a duplication of information and processes, in practical terms this means having to run the same process several times and capturing the same, or similar, information every time it is required. Furthermore, the lack of a unification criterion across these heterogeneous databases makes it difficult to support a better-structured social interaction, an electronic complement to the word of mouth interaction mechanism currently being used. The advantage of combining both, soft technology, word of mouth, paper-based billboards, newspaper-based communications and high technology, digitally mediated, is the preservation of the knowledge this community has about issues affecting them as well as the availability of an information resource fully generated by them –empowering the community. On the other hand merging and verifying this information is a major bottleneck for the definition of policies as diverse sources may host conflicting information that involve further corroboration –often expensive in terms of time, resources, and political window of opportunity.

One commonly-used alternative to overcome this heterogeneity is to apply an ontology-based framework for integration, interoperability, search and sharing of data drawn from diverse sources [1]. Such a framework normally relies on the existence of a set of mediators that address queries according to a set of local and/or global ontologies, and corresponding wrappers that overcome the heterogeneity between these models and the models from the underlying data sources.

Although there is a large amount of work devoted to mediators and to multiple types of data sources, in this paper we will focus on wrapper generation for relational databases (also known as "upgrading databases to the Semantic Web"), which is known to be a labor-intensive task. There are broadly two architectural approaches for wrapper generation that have been researched in the literature. The most common one consists in mapping a relational database schema to an existing domain ontology [2, 3, 4]. To date there has been little work automating the creation of such wrappers.

The second approach, which is the subject of the work in this paper, concerns the automatic transformation of database content to an ontological representation, normally RDF and OWL [5, 6, 7]. In this approach it is assumed that the data model entails a logical model of the application domain, and by syntactically analyzing the model's physical encoding in SQL Data Description Language (DDL) the logical model may be recovered. The current SQL standard coupled with modern software design methodology enables rich expression of domain semantics; albeit not in a form readily accessible to automated inference mechanism [8]. In addition to foreign key constraints, SQL DDL supports a variety of constraints on the range of values allowed in a table.

In this paper we describe our approach for wrapper generation, which is based on the second alternative, allowing the transformation of relational databases into OWL-DL ontologies. This approach has been used in the context of the Colombian displaced community that has been described in this introduction and that has served as a case study for our technological approach. Before describing our approach, we also provide some insights into related approaches and technologies. Later we describe how the ontology obtained from one of the analyzed schemas can be aligned with the Friend of A Friend (FOAF) ontology [9], which has facilitated the description of people within social networks (FOAF has been considered as one of the most prevalent uses of Semantic Web technology [10]).

2. RELATED WORK

While a comprehensive related work section should include not only technological or architectural options, but also a description of alternative solutions to the problem addressed in our work, we have already commented in the introduction that there are no similar systems that solve the problem of the displaced population in a Latin-American country like Colombia (and we have not found in our research any comparable technological effort for other countries). Hence our focus in this section is on the problem of addressing heterogeneity in databases following an approach where ontologies are derived from relational data sources [5, 6, 7].

Stojanovic et al. [5] provide rules for translation of relational schemas to Frame Logic and RDF Schema. This work formally defines rules for identification of classes and properties in relational schemas. However, it does not have the capability of capturing richer semantics that cannot be expressed in RDF Schema.

Li et al. [6] propose a set of rules for automatically learning an OWL ontology from a relational schema. They define the rules using a combination of some formal notation and English language. Some of their rules miss some of the semantics offered by relational schemas and some rules produce specific results for inheritance and object properties that may not accurately depict concepts across domains or database modeling choices. We believe these shortcomings are due to the lack of a formal system and thorough examination of examples capturing a variety of modeling choices in various domains.

Finally, Astrova et al. [7] provide expository rules and examples to describe a system for automatic transformation of a relational schema to OWL Full, being one of the most comprehensive approaches. However, rules are not formally defined; hence a number of transformations are ambiguous.

3. DIRECT MAPPING RELATIONAL DATABASES TO THE SEMANTIC WEB

We start this section with an example. Let us consider a relational database that captures data of displaced people in Colombia (see Table 1, an excerpt of the relational model that allows representing information about displaced people in one of the existing systems). The Person table contains data about all people in the system. The Family table lists the families that are being displaced, who is the head household member and information about the displacement. Also, a person is part of a family, and this information is recorded in the FamilyMember table. The Displacement table lists the information of a displacement: when and from where did a family leave and when and to where did a family arrive to. The City table contains a list of all cities in Colombia.

Table 1. Schema of a Displaced Population Database

```
create table PERSON {
  PERSONID integer primary key,
  NAME varchar not null,
  GENDER varchar check in (‘M’, ‘F’),
  CIVIL_STATUS varchar check in (‘Married’,
‘Single’, 'Divorced'),
  AGE integer not null}
create table FAMILY {
  FAMILYID integer primary key,
  FAMILYNAME varchar,
  HEAD integer unique not null foreign key
references PERSON(PERSONID),
  DISID integer unique not null foreign key
references DISPLACEMENT(DISID) }
create table DISPLACEMENT {
  DISID integer primary key,
  CITYFROM integer unique not null foreign key
references CITY(CITYID),
  DATEFROM date,
  CITYTO integer unique not null foreign key
references CITY(CITYID)
  DATETO date}
create table CITY {
  CITYID varchar primary key,
  NAME varchar unique not null }
create table FAMILYMEMBER {
  PERSONID integer foreign key references
PERSON(PERSONID),
  FAMILYID integer foreign key references
FAMILY(FAMILYID),
  constraint FAMILYMEMBER_PK primary key
(PERSONID, FAMILYID) }
```

In this section, we explain the transformation of a relational schema to an ontology. First we present our assumptions and explain the rationale behind them. Then, we list the predicates and functions we have defined to express transformation rules in first order logic. In the next section, we explain the transformations for data types, classes, properties and inheritance, and provide mapping tables or first order logic rules to formally define the transformations.

3.1 Assumptions

In order to translate a relational schema into an ontology, we make the following assumptions:

- The relational schema, in its most accurate form, is available in SQL DDL. Databases evolve due to changing application requirements. Such modifications are often reflected solely in the physical model, usually expressed in SQL DDL, making it the most accurate source for the structure of the database.
- The relational schema is normalized, at least up to third normal form. While all databases might not be well normalized, it is possible to automate the process of finding functional dependencies within data and to algorithmically transform a relational schema to third normal form [11, 12].

3.2 Predicates and Functions

We have defined a number of predicates and functions to aid the process of defining transformation rules in first order logic.

There are two sets of predicates in our system. RDB predicates test whether an argument (or a set of arguments) matches a construct in the domain of relational databases. Such predicates are listed below:

Rel(r)	*r* is a relation
Attr(x,r)	*x* is an attribute in relation *r*
NN(x,r)	*x* is an attribute (or a set of attributes) in relation *r* with NOT NULL constraint(s)
Unq(x,r)	*x* is an attribute (or a set of attributes) in relation *r* with UNIQUE constraint
Chk(x,r)	*x* is an attribute in relation *r* with enumerated list (CHECK IN) constraint
PK(x,r)	*x* is the (single or composite) primary key of relation *r*
FK(x,r,y,s)	*x* is a (single or composite) foreign key in relation *r* and references *y* in relation *s*
NonFK(x,r)	*x* is an attribute in relation *r* that does not participate in any foreign key

On the other hand, ontology predicates test whether an argument (or a set of arguments) matches a construct that can be represented in an OWL ontology. These predicates are:

Class(m)	*m* is a class
ObjP(p,d,r)	*p* is an object property with domain *d* and range *r*
DTP(p,d,r)	*p* is an data type property with domain *d* and range *r*
Inv(p,q)	when *p* and *q* are object properties, *p* is an inverse of *q*
FP(p)	*p* is a functional property
IFP(p)	*p* is an inverse functional property
Crd(p,m,v)	the (max and min) cardinality of property *p* for class *m* is *v*
MinC(p,m,v)	the min cardinality of property *p* for class *m* is *v*
MaxC(p,m,v)	the max cardinality of property *p* for class *m* is *v*
Subclass(m,n)	*m* is a subclass of class *n*

The constructs represented by ontology predicates are described as they appear in the rules mentioned in the upcoming sections of this paper.

We have also defined the following functions:

fkey(x,r,s)	takes a set of attributes *x*, relations *r* and *s*, and returns the foreign key defined on attributes *x* in *r* referencing *s*
type(x)	maps an attribute *x* to its suitable OWL recommended data type (we discuss data types in more detail in a later section)
list(x)	maps an attribute *x* to a list of allowed values; applicable only to attributes with a CHECK IN constraint, i.e. *Chk(x)* is true

In addition to the predicates and functions listed above, we describe the concept of a binary relation, written BinRel, as a relation that only contains two (single or composite) foreign keys that reference other relations. Such tables are used to resolve many-to-many relationships between entities. Using RDB predicates, we formally define BinRel as follows:

Rule Set 1:

$$BinRel(r,s,t) \leftarrow Rel(r) \wedge FK(q,r,\_,t) \wedge FK(p,r,\_,s) \wedge p \neq q \wedge Attr(y,r) \wedge \neg NonFK(y,r) \wedge FK(z,r,\_,u) \wedge fkey(z,r,u) \in \{fkey(p,r,s), fkey(q,r,t)\}$$

This rule states that a binary relation r between two relations s and t exists if r is a relation that has foreign keys to s and t, and r has no other foreign keys or attributes (each attribute in the relation belongs to one of the two foreign keys). Note that there is no condition that requires s and t to be different, allowing binary relations that have their domain equal to their range.

3.3 Transformation Rules and Examples

In this section we present rules and examples for transformation of a relational database to OWL ontology.

3.3.1 Producing Unique Identifiers (URIs) and Labels

Before we discuss the transformation rules, it is important to understand how we can produce identifiers and names for classes and properties that form the ontology.

The concept of globally unique identifiers is fundamental to OWL ontologies. Each class or property in the ontology must have a unique identifier, or URI. While it is possible to use the names from the relational schema to label the concepts in the ontology, it is necessary to resolve any duplications, either by producing URIs based on fully qualified names of schema elements, or by producing them randomly. In addition, for human readability, RDFS labels should be produced for each ontology element containing names of corresponding relational schema elements. Due to lack of space, we have not used fully qualified names in our examples. When needed, we append a name with an integer to make it unique, e.g. ID1, ID2 etc.

3.3.2 Transformation of Data Types

Transformations from relational schemas to ontologies require preserving data type information along with the other semantic information. OWL (and RDF) specifications recommend the use of a subset of XML Schema types [13] in Semantic Web ontologies [14, 15].

In Table 2 we present a list of commonly used SQL data types along with their corresponding XML Schema types. During transformation of data type properties, the SQL data types are transformed into the corresponding XML Schema types.

Table 2. Common SQL types and corresponding XML Schema types recommended for OWL

SQL Data Type	XML Schema Type
INTEGER	xsd:integer
FLOAT	xsd:float
BOOLEAN	xsd:Boolean
VARCHAR	xsd:string
DATE	xsd:date
TIMESTAMP	xsd:dateTime

3.3.3 Identifying Classes

According to OWL Language Guide [16], "the most basic concepts in a domain should correspond to classes …". Therefore we would expect basic entities in the data model to translate into OWL classes.

Given the definition of a binary relation, it is quite straightforward to identify OWL classes from a relational schema. Any relation that is not a binary relation can be mapped to a class in an OWL ontology, as stated in the rule below.

Rule Set 2:

$$Class(r) \leftarrow Rel(r) \wedge \neg BinRel(r,\_,\_)$$

Remember that a binary relation has exactly two foreign keys and no other attributes (see Rule Set 1). Keeping that in mind, we can see that this very simple rule covers a number of cases for identifying classes:

- All tables that do not have foreign keys should be transformed to classes. Therefore, we conclude Class(PERSON), i.e. Person should be mapped to a class since it has no foreign key. The same reasoning holds for the City table.
- All tables with one foreign key can be mapped to classes since they cannot be binary relations.
- Tables with more than two foreign keys should be transformed to classes as well. Such tables may represent an entity or an N-ary relationship between entities. Fortunately, in OWL, both the cases can be modeled the same way, i.e. by translating the entity or the N-ary relationship into a class [17].
- For tables containing exactly two foreign keys, presence of independent attributes qualifies them to be translated to classes. The table Family, with an independent attribute FamilyName, is an example, and is translated to an OWL class. The table Displacement is translated to an OWL class

Thus Rule Set 2 identifies the OWL classes from the database schema. For example:

Class(PERSON), Class(FAMILY), Class(DISPLACEMENT), Class(CITY)

3.3.4 Identifying Object Properties

An object property is a relation between instances of two classes in a particular direction. In practice, it is often useful to define object properties in both directions, creating a pair of object properties that are inverses of each other. OWL provides us the means to mark properties as inverses of each other. In our work, when we translate something to an object property, say ObjP(r,s,t), it implicitly means we have created an inverse of that property, which we write as ObjP(r',t,s).

There are two ways of extracting OWL object properties from a relational schema. One of the ways is through identification of binary relations, which represent many-to-many relationships. The following rule identifies an object property using a binary relation.

Rule Set 3:

$$ObjP(r,s,t) \leftarrow BinRel(r,s,t) \ni Rel(s) \wedge Rel(t) \wedge \neg BinRel(s,\_,\_) \wedge \neg BinRel(t,\_,\_)$$

This rule states that a binary relation r between two relations s and t, neither being a binary relation, can be translated into an OWL object property with domain s and range t. Notice that the rule implies Class(s) and Class(t) hold true, so the domain and range of the object property can be expressed in terms of corresponding OWL classes.

From our university database schema, the FamilyMember table fits the condition. FamilyMember is a binary relation between Person and Family entities, which are not binary relations. Therefore, ObjP(FAMILYMEMBER,PERSON,FAMILY) holds, and since we can create inverses, ObjP(FAMILYMEMBER',FAMILY,PERSON) and Inv(FAMILYMEMBER,FAMILYMEMBER') also hold true.

Foreign key references between tables that are not binary relations represent one-to-one and one-to-many relationships between entities. A pair of object properties that are inverses of each other and have a maximum cardinality of 1 can represent one-to-one relationships. Also, one-to-many relationships can be mapped to an object property with maximum cardinality of 1, and an inverse of that object property with no maximum cardinality restrictions.

In OWL, a property with min cardinality of 0 and max cardinality of 1 is called functional which we represent by the functor FP. If an object property is functional, then its inverse is inverse functional, represented by the functor IFP. In addition to specifying cardinality restrictions on properties in general, we can also specify such restrictions when a property is applied over a particular domain. In our rules, we use ontology predicates Crd, MinC and MaxC to specify these restrictions. The examples following the rules explain the use of these predicates.

The following rule set identifies object properties and their characteristics using foreign key references (not involving binary relations, covered in Rule Set 3) with various combinations of uniqueness and null restrictions. To simplify the rules, we first define a predicate NonBinFK that represents foreign keys not in or referencing binary relations and then express the rules in terms of this predicate.

Rule Set 4:

$$NonBinFK(x,s,y,t) \equiv FK(x,s,y,t) \ni Rel(s) \wedge Rel(t) \wedge \neg BinRel(s,\_,\_) \wedge \neg BinRel(t,\_,\_)$$

a. $ObjP(x,s,t),\ FP(x),\ MinC(x',t,0) \leftarrow NonBinFK(x,s,y,t) \wedge \neg NN(x) \wedge \neg Unq(x)$

b. $ObP(x,s,t),\ FP(x),\ Crd(x,s,1),\ MinC(x',t,0) \leftarrow NonBinFK(x,s,y,t) \wedge NN(x) \wedge \neg Unq(x)$

c. $ObjP(x,s,t),\ FP(x),\ FP(x') \leftarrow NonBinFK(x,s,y,t) \wedge \neg NN(x) \wedge Unq(x)$

d. $ObjP(x,s,t),\ FP(x),\ Crd(x,s,1),\ FP(x') \leftarrow NonBinFK(x,s,t) \wedge NN(x) \wedge Unq(x) \wedge \neg PK(x,s)$

Each rule in Rule Set 4 states that a foreign key represents an object property from the entity containing the foreign key (domain) to the referenced entity (range). Since a foreign key references at most one record (instance) of the range, the object property is functional. This entails that the inverse of that object property is inverse functional. An example is the foreign key from Study to Student which gives us: ObjP(RNO,STUDY,STUDENT), FP(RNO), Inv(RNO',RNO), IFP(RNO').

Rules 4a and 4b represent variations of one-to-many relationships.

- We can apply a stronger restriction on cardinality of the object property if the foreign key is constrained as NOT NULL. Without this constraint (rule 4a), the minimum cardinality is 0, which is covered by functional property predicate. With this constraint (rule 4b), we can set the maximum and minimum cardinality to 1.

- According to these rules, we can infer only the minimum cardinality restriction of 0 on the inverse property. Since an instance in the range could be referenced by any number of instances in the domain, we cannot apply a maximum cardinality restriction on the inverse property.

The other two rules, 4c and 4d, represent one-to-one relationships, modeled by applying a uniqueness constraint on the foreign key. It means that an instance in the range can relate to at most one object in the domain, making the inverse property functional too. This also means that the original object property is inverse functional as well.

The difference between rules 4c and 4d is that of a NOT NULL constraint that, like one-to-many relationships mentioned above, if present, gives us a stronger cardinality restriction on the object property represented by the foreign key.

Notice that none of the rules allow the foreign key to be the same as the primary key of the domain relation. Rule 4d restricts this by providing an extra condition, whereas the negation of uniqueness or NOT NULL constraints in rules 4a-c, by definition, implies this condition.

Examples of object properties and their characteristics obtained from the relational schema by applying Rule Sets 3 and 4 are:

ObjP(FAMILYMEMBER,PERSON,FAMILY),
ObjP(FAMILYMEMBER',FAMILY,PERSON),
Inv(FAMILYMEMBER, FAMILYMEMBER')
ObjP(CITYFROM,DISPLACEMENT,CITY),
FP(CITYFROM), IFP(CITYFROM'),
MinC(CITYFROM',CITY,0)
ObjP(CITYTO,DISPLACEMENT,CITY),
FP(CITYTO), IFP(CITYTO'), MinC(CITYTO',CITY,0)
ObjP(HEAD,FAMILY,PERSON), FP(HEAD), FP(HEAD'),
Crd(HEAD,FAMILY,1)
ObjP(DISID,FAMILY,DISPLACEMENT),
FP(DISID), FP(DISID'), Crd(DISID,FAMILY,1)

3.3.5 Identifying Data Type Properties

Data type properties are relations between instances of classes with RDF literals and XML Schema data types. Like object properties, data type properties can also be functional, and can be specified with cardinality restrictions. However, unlike object properties, OWL DL does not allow them or their inverses to be inverse functional.

Attributes of relations in a database schema can be mapped to data type properties in the corresponding OWL ontology. Rule Set 5 identifies data type properties.

Rule Set 5:

a. $DTP(x,r,type(x)), FP(x) \leftarrow NonFK(x,r)$

b. $DTP(x,r,type(x)), FP(x), Crd(x,r,1) \leftarrow NonFK(x,r) \ni NN(x,r)$

c. $DTP(x,r,type(x) \cap list(x)), FP(x) \leftarrow NonFK(x,r) \ni Chk(x,r)$

Rule Set 5 says that attributes that do not contribute towards foreign keys can be mapped to data type properties with range equal to their mapped OWL type. Since each record can have at most one value per attribute, each data type property can be marked as a functional property. When an attribute has a NOT NULL constraint, rule 5b allows us to put an additional cardinality restriction on the property. Rule 5c allows us to infer stronger range restrictions on attributes with enumerated list (CHECK IN) constraints.

In some cases, it may be possible to apply more than one rule to an attribute. In such cases, all possible rules should be applied to extract more semantics out of the relational schema. Some data type properties extracted from our sample university database schema are:

DTP(PERSONID,PERSON,xsd:integer), FP(PERSONID), Crd(PERSONID,PERSON,1)
DTP(GENDER, PERSON, xsd:string?{M,F}), FP(GENDER)
DTP(NAME,PERSON,xsd:string), FP(NAME), Crd(NAME,PERSON,1)

3.3.6 Identifying Inheritance

Inheritance allows us to form new classes using already defined classes. It relates a more specific class to a more general one using subclass relationships [16].

Inheritance relationships between entities in a relational schema can be modeled in a variety of ways. Since most of these models are not limited to expressing inheritance alone, it is hard to identify subclass relationships.

The following rule describes a special case that can be used only for inheritance modeling in a normalized database design.

Rule Set 6:

$Subclass(r,s) \leftarrow Rel(r) \wedge Rel(s) \wedge PK(x,r) \wedge FK(x,r,\_,s)$

This rule states that an entity represented by a relation *r* is a subclass of an entity represented by relation *s*, if the primary key of *r* is a foreign key to *s*.

4. Integration with FOAF

As a result of direct mapping, an ontology is derived from the relational database schema. Conversely, the expressiveness of this ontology depends on the structure of the relational database schema. If the schema has been developed with sophisticated tools and in a normalized manner, the schema can portray enough semantics to create a semantically rich ontology. If this is the case, then our proposal is the fastest way to create RDF from relational database data, according to an automatically-derived ontology.

Furthermore, in our case study, we are mostly interested in data about people and about people's interaction. Relational DBs for displaced population normally store this type of information in a similar way to FOAF; hence we propose to map the database-derived ontology to the FOAF ontology manually, by replacing the concepts and properties obtained with those available in FOAF. This will enable our initial objective of facilitating social networks for the displaced population, since FOAF is widely used and social networking FOAF-based tools are available.

As an example of this transformation, the "PERSON" class and its properties would be equivalent to the foaf:Person class and its respective properties; the "FAMILYMEMBER" object property may be considered as a specialization of the foaf:knows property, etc. Obviously, depending on the source used to derive the ontology, different mappings can be generated. For example, if one was to consider replacing all the information about people with FOAF, but continuing using other parts of the database-derived ontology, one could still use a SPARQL query like the ones shown in Table 3 and Table 4.

Table 3. SPARQL query to the database-derived ontology: show the family and the members of the family who have been displaced from „San Jose del Guaviare" to „Bogota"

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX :
<http://www.example.com/displaced.owl#>

SELECT  ?family ?firstName ?familyName
WHERE {
     ?family a :Family .
     ?y :isFamilyMember ?family .
     ?y foaf:firstName ?firstName .
     ?y foaf:family_name ?familyName .
     ?family :hasDisplacement ?x .
     ?x :comesFrom ?cityFrom .
     ?x :sendsToCity ?cityTo .
     ?cityFrom :hasCityName "San Jose del
Guaviare" .
     ?cityTo :hasCityName "Bogota" .
}
```

Table 4. SPARQL query to the database-derived ontology mapped to FOAF: show all the people and their current location

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT  ?firstName ?familyName ?place
WHERE {
     ?x a foaf:Person .
    ?x foaf:firstName ?firstName .
    ?x foaf:family_name ?familyName .
    ?x foaf:based_near ?place
```

```
}
```

Our main assumption is that by having several data sources transformed into a Semantic Web representation (RDF for data and OWL for the ontologies), interoperability can be achieved more easily, since format heterogeneity can be easily overcome. Moreover, as a direct consequence of having several sources being aligned with the same set of ontologies (FOAF in our case), semantic interoperability is further improved. In our case we have also proposed to extend the FOAF ontology so that it can represent displaced population.

5. CONCLUSIONS

SQL DDL is a standard for representing the physical schema of applications that use relational databases. Although SQL DDL is not a knowledge representation language, it is capable of capturing some semantics of the application domain. We have defined a system for automatic transformation of normalized SQL DDL schemas into a OWL ontology. We have defined our entire set of transformation rules in first order logic eliminating syntactic and semantic ambiguities and allowing for easy implementation of the system in languages like JESS or Datalog.

Once an ontology is obtained for a domain, previously represented by a relational schema, the actual database content can be easily translated into its corresponding RDF representation, and then alignments between the obtained model and other existing ontologies can be achieved. Being this similar to the work done in traditional local as view approaches.

Our case study illustrates how this approach can be applied within the context of displaced population; more specifically in the case of the Colombian displaced population. We will continue to work on this scenario by incorporating several of the heterogeneous public and private data sources about displacement. The more coherently integrated data, the more useful information there will be available for policy makers and also for the displaced population. Social networks within this context don't just allow people with similar interests to know about each other; more importantly, in this scenario social networks play an important role by facilitating the reconstruction of the social tissue that was lost because of the social conflict. Such specialization, extension, of FOAF opens a new window for the semantic web and social networks to be fully tested and used.

6. ACKNOWLEDGMENTS

Special thanks to Carlos Castro for his useful comments regarding displaced population. This research was funded in part by the National Science Foundation Grant IIS-0531767.

7. REFERENCES

[1] Wache, H., T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. "Ontology-based integration of information - a survey of existing approaches." IJCAI-01 Workshop: Ontologies and Information Sharing. Ed. H. Stuckenschmidt 2001, 108-117.

[2] An, Y., Borgida, A., and Mylopoulos, J. Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences. In Proceedings of On The Move to Meaningful Internet Systems, 2005.

[3] Bizer, C., and Seaborne, A. D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs. In Poster Proceedings of 3rd International Semantic Web Conference, 2004.

[4] Barrasa, J., Corcho, O. and Gomez-Perez, A. R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. A Second Workshop on Semantic Web and Databases (SWDB), 2004.

[5] Stojanovic, L., Stojanovic, N., and Volz, R. Migrating data-intensive web sites into the semantic web. In Proceedings of the ACM Symposium on Applied Computing, 2002.

[6] Li, M., Du, X., and Wang, S. Learning ontology from relational database. In Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, 2005.

[7] Astrova, I., Korda, N., and Kalja, A. Rule-Based Transformation of SQL Relational Databases to OWL Ontologies. In Proceedings of the 2nd International Conference on Metadata & Semantics Research, October 2007.

[8] Sequeda, J.F., Tirmizi, S.H., and Miranker, D.P. SQL Databases are a Moving Target. Position Paper for W3C Workshop on RDF Access to Relational Databases, 2007.

[9] Friend-of-a-Friend (FOAF). http://www.foaf-project.org

[10] Ding, L., and Finin, T. Characterizing the Semantic Web on the Web. In Proceedings of the 5th International Semantic Web Conference. 2006

[11] Du, H., and Wery, L. Micro: A normalization tool for relational database engineers. Journal of Network and Computer Applications, vol. 22, no. 4, pp 215-232, 1999.

[12] Wang, S., Shen, J., and Hong, T. Mining fuzzy functional dependencies from quantitative data. IEEE International Conference on Systems, Man and Cybernetics, 2000.

[13] Biron, P.V., Permanente, K., Malhotra, A. (eds.). XML Schema Part 2: Datatypes Second Edition. W3C Recommendation, http://www.w3.org/TR/xmlschema-2/ (11/24/2008).

[14] Dean, M and Schreiber, G. (eds.). OWL Web Ontology Language Reference. W3C Recommendation, http://www.w3.org/TR/owl-ref/ (11/24/2008).

[15] Hayes, P. (ed.). RDF Semantics. W3C Recommendation, http://www.w3.org/TR/rdf-mt/ (11/24/2008).

[16] Smith, M.K., Welty, C., and McGuinness, D.L. (eds.). OWL Web Ontology Language Guide. W3C Recommendation, http://www.w3.org/TR/owl-guide/ (11/24/2008).

[17] Noy, N., and Rector, A. (eds.). Defining N-ary Relations on the Semantic Web. W3C Working Group Note. http://www.w3.org/TR/swbp-n-aryRelations/ (11/24/2008).