The Knowledge Engineering Review, Vol. 26:4, 445–486. © Cambridge University Press, 2011
doi:10.1017/S0269888911000208

Survey of directly mapping SQL databases to the Semantic Web

JUAN F. SEQUEDA[1], SYED HAMID TIRMIZI[1], OSCAR CORCHO[2] and DANIEL P. MIRANKER[1]

[1]*Department of Computer Sciences, The University of Texas at Austin, Austin, TX, USA;*
e-mail: jsequeda@cs.utexas.edu, hamid@cs.utexas.edu, miranker@cs.utexas.edu
[2]*Ontology Engineering Group, Universidad Politecnica de Madrid, Madrid, Spain;*
e-mail: ocorcho@fi.upm.es

Abstract

The Semantic Web anticipates integrated access to a large number of information sources on the Internet represented as Resource Description Framework (RDF). Given the large number of websites that are backed by SQL databases, methods that automate the translation of those databases to RDF are crucial. One approach, taken by a number of researchers, is to directly map the SQL schema to an equivalent Web Ontology Language (OWL) or RDF Schema representation, which in turn, implies an RDF representation for the relational data. This paper reviews this research, and derives a consolidated, overarching set of translation rules expressible as a stratified Datalog program. We present all the possible key combinations in an SQL schema and consider their implied semantic properties. We review the approaches and characterize them with respect to the scope of their coverage of SQL constructs.

1 Introduction

An objective of the Semantic Web is to enable web-wide data integration. The Semantic Web architecture entails a standardized data model, which encodes a directed labeled graph (Resource Description Framework (RDF); Klyne & Carroll, 2004), adding metadata encoded as an ontology (in RDF Schema (RDFS) or Web Ontology Language (OWL); Brickley & Guha, 2004; Smith *et al.*, 2004) and linking the data to other data across the web (Linked Data; Bizer *et al.*, 2009). A standardized query language and service protocol (SPARQL Protocol and RDF Query Language (SPARQL); Prud'hommeaux & Seaborne, 2008) enables the web to appear as a giant database and allow distributed execution of queries (Hartig *et al.*, 2009).

Broadly stated, there are two architectural approaches to integrating relational databases with the Semantic Web. In the first approach, called *direct mapping*, which is the subject of this survey, the database's SQL schema is mechanically transformed into an ontology and the relational data is exposed as instances of the generated ontology (Figure 1; Stojanovic *et al.*, 2002a, 2002b; Astrova, 2004; Buccella *et al.*, 2004; Li *et al.*, 2005; Shen *et al.*, 2006; Astrova *et al.*, 2007; Tirmizi *et al.*, 2008). The second approach assumes there is an existing domain ontology and aims to *wrap* the database such that its contents appears as instances of that ontology. Wrapping systems typically provide a mapping language used to detail the correspondence of database schema names to ontology concepts and properties. The ontology may be either a global or local ontology (Figure 2; Collet *et al.*, 1991; Arens *et al.*, 1993; Barrasa *et al.*, 2004; Bizer & Seaborne, 2004; Korotkiy & Top, 2004; Svihla & Jelinek, 2004; An *et al.*, 2005, 2006; Barrasa & Gomez-Perez, 2006; Chen *et al.*, 2006; de Laborda & Conrad, 2005; Duo *et al.*, 2006; Laclavik, 2006; Xu *et al.*, 2006). A standard for both approaches is in process, sponsored by the World Wide Web Consortium (W3C; Arenas *et al.*, 2010; Das *et al.*, 2010).

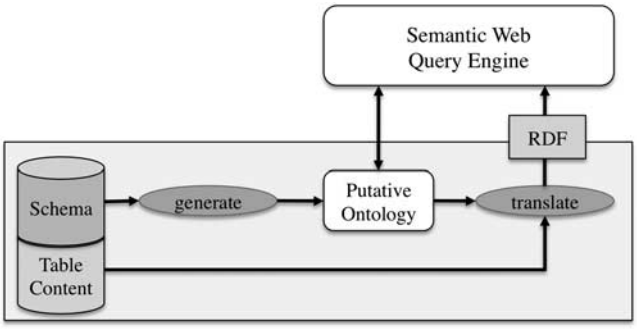

Figure 1 Direct Mapping of a Relational Database to the Semantic Web

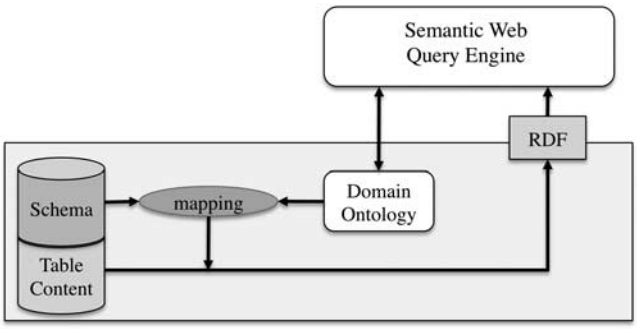

Figure 2 Relational Database to Ontology Mapping

We distinguish the starting point of a direct mapping as an *SQL Schema*, rather than a relational schema, and the result of the mapping is a *putative ontology*. Formal definition and in some common usage, '*relational schema*' means only the names of a database's tables and their columns. However, in the SQL Data Definition Language (SQL-DDL), in addition to defining the table and column names, this language can also define the database's primary and foreign key constraints, as well as column and table constraints (e.g. age >0; Garcia-Molina *et al.*, 2009). It is by virtue of these SQL extensions of the relational model that an SQL database can encode domain semantics (Chiang *et al.*, 1994). For example, foreign key constraints express one-to-many mappings and column constraints that may capture valuable range restrictions (e.g. $0 \leqslant$ angle $\leqslant 360$). Further, starting with SQL99, SQL-DDL includes views and triggers. Although not yet addressed in direct mapping efforts, these database language constructs are derived from forward and backward rule systems (Stonebraker, 1986).

The target of direct mapping includes, RDFS and OWL and its variants. Although the variation in power, and debate among these languages as ontology languages contribute to naming the result of a direct mapping, a *putative ontology*, it is the high variability of the quality of an individual SQL schema as a domain model that calls for hedging on the quality of the resulting ontology. It is common, due to either historical reasons or engineering methodology, for a legacy SQL schema to express no more details than a relational schema. If constraints are not explicit, a direct mapping will yield few, if any, semantic details of the domain. However, we and others have observed that if a database application was developed with attention to Entity-Relationship or Unified Modeling Language (UML) modeling methods, the quality of a putative ontology may replicate the quality of initial logical data model (Lubyte & Tessaris, 2009). UML models are often the basis of ontology representation (Kogut *et al.*, 2002).

A criticism of the direct mapping approach is that putative ontology will still require integration with some global domain ontology if larger-scale interoperability is sought. In response we note,

He *et al*. (2007) determined that Internet accessible databases contained up to 500 times more data compared with the static Web and roughly 70% of websites are backed by relational databases. Thus, the number of websites suggests that it is paramount to develop automatic methods for integrating relational databases with the Semantic Web. Direct mapping methods are intrinsically automatic. Further, direct mapping shifts the mapping problem from a relational schema-to-ontology matching problem to an ontology-to-ontology matching problem. Even in isolation of relational databases, the successful creation of the Semantic Web will see the development of successful ontology-to-ontology mappings (Vrandecic & Sure, 2008). Thus, if the Semantic Web enjoys larger success, direct mapping of relational databases will be sufficient. One can also anticipate that matching putative ontologies to domain ontologies as a specialized and easier instance of ontology-to-ontology mapping (Sequeda *et al*., 2008).

The following examples introduce how SQL-DDL may encode domain semantics, but naively generalizing from examples may lead to errors. We use a University's relational database as a running example. The schema defines table structures concerning students, their enrollment, and courses being offered by a specific department. Instances of the database are illustrated in Figure 3.

A central focus of direct mapping efforts concerns the semantic interpretation of the relationships among tables formed between primary keys and foreign keys. Examining the *COURSE* and *DEPT* tables, first in Figure 3, one may construe from the values in the *DEPT* and *DID* columns that both courses in the example are Computer Science courses. The SQL statements in Figure 4 assert that the values of those columns encode, precisely, a one-to-many relationship. An equivalent ontological representation is also illustrated in Figure 4, representing a semantic assertion that '*a course is offered by a department*'.

The SQL-DDL in Figure 4 is just one of at least three SQL declarations for the tables at hand. Note, in the ontology diagram, the relationship between *COURSE* and *DEPT* has been named *offeredBy*, but that name does not appear in the SQL (the name of the constraint that appears is *dept_fk*). As in the illustrated SQL syntax, foreign key constraint definitions may be embedded in a CREATE TABLE statement. However, such constraints may also be declared using an ADD CONSTRAINT command. If the constraint would have been named *offeredBy* then a direct

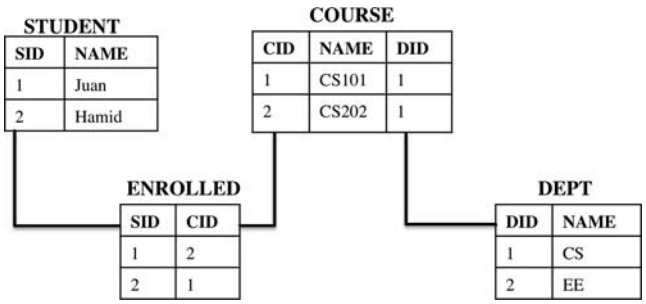

Figure 3 Example of University's relational database

```
CREATE TABLE COURSE{
 CID INT PRIMARY KEY,
 NAME VARCHAR NOT NULL,
 DEPT INT,
 CONSTRAINT DEPT_FK FOREIGN KEY(DEPT)
REFERENCES DEPT(DID)}

CREATE TABLE DEPT{
 DID INT PRIMARY KEY,
 NAME VARCHAR NOT NULL}
```

Figure 4 SQL-DDL and Ontology for Course and Dept

```
CREATE TABLE STUDENT{
 ID_STUDENT INT,
 ID_PERSON INT,
 DEGREE_STUDENT VARCHAR NOT NULL,
 CONSTRAINT PERSON_FK FOREIGN
KEY(ID_PERSON) REFERENCES
PERSON(ID_PERSON)
 PRIMARY KEY(ID_STUDENT, ID_PERSON)

CREATE TABLE PERSON{
 ID_PERSON INT PRIMARY KEY,
 NAME_PERSON VARCHAR NOT NULL}
```

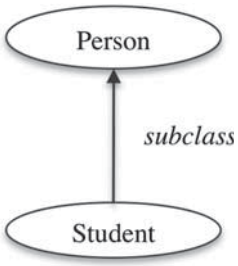

Figure 5 SQL-DDL and Ontology for Person and Student

```
CREATE TABLE ORDER{
 ID_ORDER INT PRIMARY KEY,
 ORDER_NAME VARCHAR NOT NULL}

CREATE TABLE ORDERITEM{
 ID_ORDER INT,
 ID_ITEM INT,
 ITEM_NAME VARCHAR NOT NULL,
 CONSTRAINT PERSON_FK FOREIGN
KEY(ID_ORDER) REFERENCES ORDER(ID_ORDER)
 PRIMARY KEY(ID_ORDER, ID_ITEM)
```

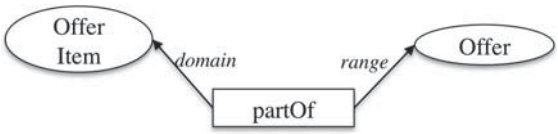

Figure 6 SQL-DDL and Ontology for Order and Order Item

mapping system could accurately label the relationship between *COURSE* and *DEPT*. In a third form of declaration, no constraints may be declared. Any relationship between the *COURSE* and *DEPT* tables must be deduced.

In SQL, keys may be simple, or compound, that is a single column or multiple columns. For example, in Figure 3 each row in table *ENROLLED* holds a pair of foreign keys, one acting as a surrogate for a *STUDENT*, the other a surrogate for a *CLASS*. Thus, the table encodes a many-to-many binary (2-ary) relationship representing which classes were taken by which students. In such tables, it is common to declare the pair of foreign keys to be the primary key of the table; that is a compound primary key is composed of two simple foreign keys.

In this survey we assess the prior work, in part, by first determining that keys may appear in 10 different syntactic cases (see Section 4). We anticipate that each syntactic type may encode, and distinguish, a particular semantic relationship. Thus, as the mapping rules of each system are considered, they are categorized according to these 10 syntactic cases.

A challenge is that many of the 10 different categories of key structure do not represent a unique semantic association. Nowhere is this more troublesome than in the encoding of inheritance.

Figure 5 illustrates an example representing inheritance between *STUDENT* and *PERSON* using a Foreign Key as a subset of a Primary Key.

In this case, the table *PERSON* becomes the class Person and the table *STUDENT* becomes a class Student, which is a subclass of the class Person. Due to the fact that one of the primary keys of the table *STUDENT* is also a foreign key to the table *PERSON*, a user can consider this is an inheritance relationship. Nevertheless, this same SQL schema, which represents inheritance in this example, may model a different type of relationship: *part of* as shown in Figure 6.

The previous example models the relationship between an *ORDERITEM*, which is part of an *ORDER*, which does not represent inheritance. Thus, it is easy to make mistakes while thinking that a specific SQL schema may always represent the same semantics.

When viewed together, the papers reviewed in this survey enumerate a large number of alternatives to interpret the same SQL schemas. Building on these, we present a consolidated direct mapping system (Section 6). The consolidated system covers all possible associations formed by foreign key relationships.

1.1 Scope, contributions and organization of this survey

The objective of this survey is to present in a single place the work the most relevant approaches on the direct mapping of SQL schemas to an ontology. The contributions of this survey are:

- To define a generic mapping of components of SQL-DDL to layers of the Semantic Web layer cake, by looking into the evolution of SQL and understanding how domain semantics can be encoded in SQL (Section 2).
- To present transformation patterns between relational database schemas and ontologies (Section 3).
- A theorem that describes all the possible combination of foreign keys and primary keys, of which are 10, in order to cover the entire range of possible relationships that can be described in an SQL schema (Section 4).
- To compare seven relevant direct mapping approaches characterized by the target languages, four source constructs (Table definition, Attribute and Constraints, Inheritance and n-ary Relationships) and the coverage of the 10 cases from our theorem (Section 5).
- A consolidated direct mapping system, which builds on the surveyed efforts and satisfies all of the 10 cases of primary key and foreign key combination (Section 6).

The work presented in this paper has been the foundation of the developing W3C standard (Arenas *et al*., 2010).

2 SQL data definition language and the Semantic Web languages

SQL standards have been evolving since first proposed in 1986. In each of five versions, SQL 86–89, 92, 99, 2003, 2008, the SQL language standard increases in the expressive functionality (Pratt 1990, 1995; SQL3; Sequeda *et al*., 2007). It is arguable that the first standard SQL language was not expressive enough to explicitly encode the database's application semantics. Initially the development of the relational model was largely concerned with the representation of data (Figure 7(a)) as n-ary tuples and the correctness of syntactic transformations leveraging relational algebra (Meier, 1983). A central concern was the theory of normal forms; that is the identification of functional dependencies among columns, the concomitant identification of keys and the factoring of tables to reduce and eliminate redundant storage of data (Garcia-Molina *et al*., 2009). In 1986 the prospect of the database management system enforcing constraints had not reached commercial interest.

Evident in the sequence of SQL standards are increasingly powerful elements for capturing application domain semantics. These elements include consistency constraints, database views and triggers. The inclusion of trigger definitions and view definitions in SQL was initially inspired by knowledge-base systems that capture and execute both forward and backward rule systems (Stonebraker, 1986). This is not to say that SQL-DDL is a knowledge representation language; the semantics of most of its constructs are not directly comparable with constructs of an ontology language. However, SQL-DDL is capable of capturing some semantics of the domain modeled in

a) Relational Model Before SQL	Employee (name, age)
b) Table Definition SQL 86-89	CREATE TABLE employee (name VARCHAR(100), age INTEGER)
c) Constraints SQL 92	CREATE TABLE employee(name VARCHAR(100) PRIMARY KEY, salary INTEGER NOT NULL, type CHAR(8) CHECK(type IN ('TEMP','FULLTIME','CONTRACT')) dept_name FOREIGN KEY (dept) REFERENCES department (name)) CREATE TABLE department(name VARCHAR(100) PRIMARY KEY)
d) Triggers SQL 99	CREATE TRIGGER sal_adjustment AFTER UPDATE OF salary ON employee REFERENCING OLD AS OLD_EMP NEW AS NEW_EMP FOR EACH ROW WHEN (NEW_EMP.SALARY > (OLD_EMP.SALARY *1.20)) BEGIN ATOMIC SIGNAL SQLSTATE '75001'('Invalid Increase: > 20%'); END

Figure 7 Evolution of SQL-DDL

an SQL application. Such semantics can be extracted from the schema by identifying grammatical structure (Chiang *et al.*, 1994).

2.1 The evolution of SQL data models

SQL is organized in three parts. The SQL-DDL, the SQL Data Manipulation Language (SQL-DML) and the SQL Data Query Language (SQL-DQL). The SQL-DDL is used to define both the logical and physical representation of data in a database. We provide an overview of the SQL-DDL features as they have evolved and show their parallel to the expressive hierarchy of Semantic Web languages. Taking just some literary license, we can structure the chronological development of increasingly powerful SQL dialects as layers of expressiveness that correspond precisely to the Semantic Web layer cake.

For the purpose of this paper, the following succinct description of the SQL86-89 DDL suffices; starting from relational algebra, a standard set of data types were defined, abbreviated formal relational notations were replaced by full English words (SELECT, CREATE TABLE, etc.), and the now familiar accouterments of human readable computer languages added (Figure 7(b)). The often-heard claim that SQL databases do not provide for the encoding of data semantics is apt for this version of SQL (Pratt, 1990).

SQL 92 added data integrity constraints: CHECK, PRIMARY KEY, FOREIGN KEY and UNIQUE (Pratt, 1995). This first extension already enables database administrators to encode application semantics by connecting tables and permitting us to know beforehand what values are allowed. In conjunction with the basic table definition of SQL 89, this version of SQL is the one in common use today (Figure 7(c)).

SQL 99, also known as SQL3, saw the addition of Triggers. Triggers are forward-chaining rules whose predicates are conditioned on changes to table content. Triggers are often used to maintain correctness and to implement heterogeneous data integration (Ceri & Widom, 1993). Figure 7(d) is a trigger that maintains an invariant on the database concerning salary inversion. SQL 2003 introduces XML (extensible markup language)-related features. We find, so far, that the XML-related extensions enable only the encoding of syntactic properties specific to XML.

2.2 SQL and the Semantic Web layer cake

Given the context of the Semantic Web stack, the analysis of the SQL-DDL evolution suggests a correspondence between individual SQL definitions and the layers in the Semantic Web stack (Figure 8). A similar, explicit mapping of layers was also done in an effort to integrate ontologies expressed in the Open Biomedical Ontology (OBO) language and ontologies expressed in OWL (Tirmizi *et al.*, 2011). We anticipate that the organization of the Semantic Web with respect to the assignment of expressive power to each of its respective layers embodies a larger organizing principal.

We observe that direct mapping systems have different goals with respect to how much of the expressive power of SQL-DDL is considered. Similarly, the target language differs among projects.

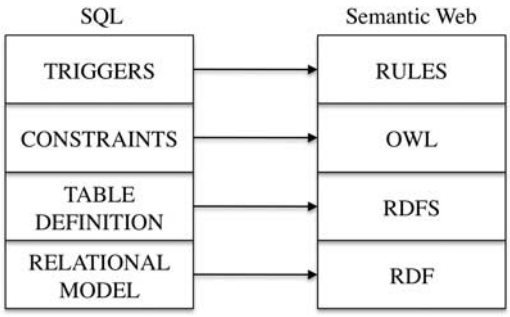

Figure 8 Layer Cake Correspondence between SQL and the Semantic Web

We begin by demonstrating how implicit domain semantics expressed in SQL-DDL can be made explicit. The examples are drawn from the University SQL schema and its corresponding ontology, detailed in Section 2.3. We continue to discuss incompatibilities between relational databases and ontologies.

2.2.1 Relational model and Resource Description Framework

The relational model expresses the syntactic structure of the data stored in a relational database. Thus the mapping targets the Semantic Web's data model: RDF (Klyne & Carroll, 2004). To export the content in a relational database to the Semantic Web, it is necessary to create an RDF representation of the relational data. RDF expresses data in triples as a labeled graph in the form of a Subject–Predicate–Object statement. The n-tuples can be expressed as a series of n RDF statements. In the bottom part of Figure 9, we show how the n-tuple *Employee(Juan, 21)* can be represented as an RDF graph. In RDF, the subject of a triple must be an identifier, which we represent as *T1* in the example.

2.2.2 Table definition and Resource Description Framework Schema

SQL89 introduced the table definition, which creates an SQL schema for the relational data. Each column has a specific data type. Similarly, RDFS defines a schema for RDF data, which creates classes and properties (Brickley & Guha, 2004). By using this version of SQL, its domain semantics can be mapped to the RDFS layer. RDFS can represent the table definition (Figure 7b) by having a class *Employee* with properties *name* and *age*. In this example, the column name *age* becomes a property with domain *Employee* and range *int*, as shown in the upper part of Figure 9.

2.2.3 Constraints and Web Ontology Language

OWL provides more expressivity than RDFS (Smith *et al*., 2004). For example, in OWL, properties can be defined as Object properties or Datatype properties. The former has a class as its domain and range whereas for the latter, the domain is a class and the range is a datatype. Additionally, properties can be classified as Symmetric, Transitive, Functional or Inverse Functional. Moreover, OWL allows adding cardinalities and quantifications.

As SQL matured, it contains new components that offer more domain semantics such as *PRIMARY KEY*, *FOREIGN KEY*, *NOT NULL*, *UNIQUE* and *CHECK*. For example, a primary key indicates that each row has a unique identifier and a check constraint can limit the values that an attribute can have. These semantics are richer than the basic table definition and therefore should be mapped to a higher level in the Semantic Web layer cake. We have observed that the use of SQL with all the possible constraints will map directly to OWL (hence the use of OWL class instead of RDFS class).

Following the example in Figure 7c, the schema and its constraints can be represented in OWL. As a general assumption, a table can be considered an OWL class except if it represents a binary (2-ary) relationship, therefore it is an OWL Object property. All attributes that are referential constraints are OWL Object properties. Their domain is the current table and range is the referenced table. Attributes that are not referential constraints are OWL Datatype properties.

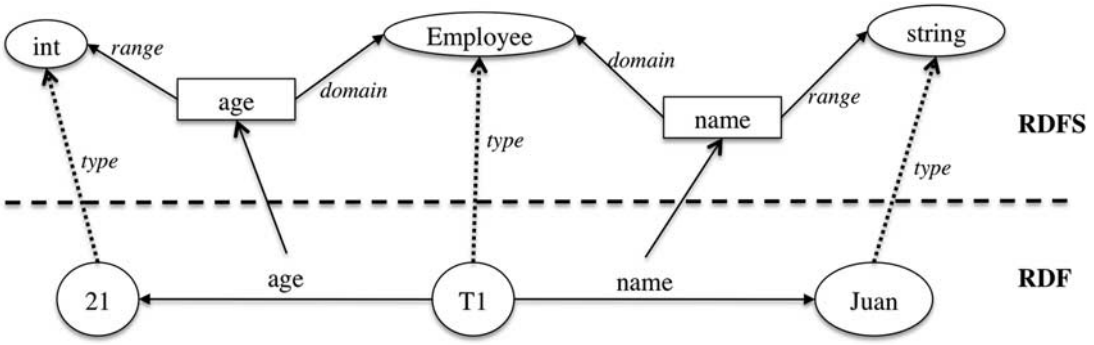

Figure 9 Resource Description Framework (RDF) and RDF Schema (RDFS) example

University Database Schema

```
create table PERSON {
ID integer primary key,
NAME varchar not null }
create table STUDENT {
  ROLLNO integer primary key,
  DEGREE varchar,
  ID integer unique not null foreign key references PERSON(ID)}
create table PROFESSOR {
  ID integer primary key,
  TITLE varchar,
  constraint PERSON_FK foreign key (ID) references PERSON(ID)}
create table DEPT {
  CODE varchar primary key,
  NAME varchar unique not null }
create table SEMESTER {
  SNO integer primary key,
  YEAR date not null,
  SESSION varchar check in ('SPRING', 'SUMMER', 'FALL')}
create table COURSE {
  CNO integer primary key,
  TITLE varchar,
  CODE varchar not null foreign key references DEPT(CODE)}
create table OFFER {
  ONO integer primary key,
  CNO integer foreign key references COURSE(CNO),
  SNO integer foreign key references SEMESTER(SNO),
  PID integer foreign key references PROFESSOR(ID),
  CONO integer foreign key references OFFER(ONO)}
create table STUDY {
  ONO integer foreign key references OFFER(ONO),
  RNO integer foreign key references STUDENT(ROLLNO),
  GRADE varchar,
  constraint STUDY_PK primary key (ONO, RNO)}
create table REG {
  SID integer foreign key references STUDENT(ID),
  SNO integer foreign key references SEMESTER(SNO),
  constraint REG_PK primary key (SID, SNO)
}
```

Figure 10 Schema of a University Database

2.2.4 Triggers and rules

SQL99 introduces a new semantic component: triggers, as shown in Figure 7d. This new component can be considered as business rules that guarantee the integrity of data in a relational database and derive new information materializing it. Even though these rules are not used for inference, they may expose rules about the domain. For example, the trigger in Figure 7d explains that the maximum salary increase is of 20%. Triggers like the previous one may be analyzed to extract the rules and transform them to Semantic Web format. However, no direct mapping approach integrates triggers in the transformation process. The integration of triggers into direct mapping is still an open research question.

2.3 An example of direct mapping transformation

Consider the SQL schema for a university, as shown in Figure 10.

The *PERSON* table contains data about all the people, some of them may be students and present in the *STUDENT* table, and some may be professors and present in the *PROFESSOR* table. The *DEPT* table lists the departments in the university where each department has a unique

Domain Expert's Ontology

```
Ontology(<urn:sql2owl>

 ObjectProperty(<REG> domain(<STUDENT>) range(<SEMESTER>))
 ObjectProperty(<REG_I> inverseOf(<REG>))
 ObjectProperty(<COURSE.DEPTCODE> Functional domain(<COURSE>) range(<DEPT>))
 ObjectProperty(<COURSE.DEPTCODE_I> InverseFunctional inverseOf(<COURSE.DEPTCODE>))
 ObjectProperty(<OFFER.CONO> Transitive Symmetric domain(<OFFER>) range(<OFFER>))
 ...

 DatatypeProperty(<COURSE.CNO> Functional domain(<COURSE>) range(xsd:integer))
 DatatypeProperty(<SEMESTER.YEAR> Functional domain(<SEMESTER>) range(xsd:date))
 DatatypeProperty(<SEMESTER.SESSION> Functional domain(<SEMESTER>)
  range(oneOf("SPRING" "SUMMER" "FALL")) range(xsd:string))
 ...

 Class(<PERSON> partial ...)
 Class(<PROFESSOR> partial <PERSON> ...)
 Class(<STUDENT> partial <PERSON>
  restriction(<STUDY.RNO_I> minCardinality(0)) ...)
 Class(<COURSE> partial restriction(<COURSE.DEPTCODE> cardinality(1))
  restriction(<COURSE.CNO> cardinality(1)) ...)
 ...
)
```

Figure 11 Parts of an ontology corresponding to the SQL schema in Figure 10, developed by a domain expert. The ontology is presented in Web Ontology Language Abstract Syntax

name, and the *COURSE* table lists the courses for every department. The *SEMESTER* table contains a list of semesters, which have a year and one of the three seasons, Spring, Summer or Fall, associated with them. A course could be offered in a particular semester with a particular professor, and recorded in the *OFFER* table.

Two offered courses could be co-offered, and recorded as a self-relation in the *OFFER* table. A student could study an offered course, which is recorded in *STUDY* table. Also, a student could be registered in a semester with or without taking a course, and this information is recorded in the *REG* table.

For a domain expert, it is easy to recognize the concepts in this SQL schema and to identify the semantics of their properties and different kinds of relationships that exist between these concepts. Figure 11 shows an ontology analogous to the given SQL schema, developed by a domain expert.

It is apparent that the ontology does not correspond exactly to an SQL schema. Ontologies can and often do encode domain semantics that cannot be encoded in SQL-DDL. Therefore, a translation of an SQL schema to an ontology will suffer of ambiguity, and several disparities, which are explained in the following section.

3 Transformation patterns between relational databases and ontologies

Because relational databases and ontologies are motivated by different requirements, it is reasonable to expect some disparities among them, even in terms of basic assumptions. It is important to understand the mismatches and to make educated choices when confronted with such problems. Below, we discuss some key issues that affect a direct transformation system. We show an example of how different database normalizations can affect the putative ontology that is being derived from the SQL schema. It follows by a discussion on the difficulties to identify inheritance and property

Table 1 Summary of Relational Database direct mapping to RDFS and OWL

Relational Database	RDFS Ontology	OWL Ontology
Non-binary Table	RDFS Class	OWL Class
Binary Table	RDFS Property	OWL Object Property
Column	RDFS Property	OWL Datatype Property
Foreign Key	RDFS Property	OWL Object Property

RDFS = Resource Description Framework Schema; OWL = Web Ontology Language.

Table 2 Example of a simple database for Books

Books					
ISBN	Title	Publisher	City	Subject	Author name
123-456	ABC	Publisher XYZ	Dallas	Science Fiction	John Smith
321-654	DEF	Publisher XYZ	Dallas	Science Fiction	Jane Doe
987-654	XYZ	Publisher ABC	Austin	Auto-biography	Jane Doe

characteristics in relational schemas, followed by the effects of identifying different types of relationships and constraints. Then we discuss the effect of the open world (OW) assumption in ontologies, when a relational database with a closed world (CW) assumption is translated into an ontology.

3.1 Basic transformations

Several components of an SQL schema can be seen as obvious transformations to an ontology. In most cases, a table is mapped to an ontological class unless it is a table that represents a binary relationship. In this case, that table would be mapped to a property that connects two ontological classes. A foreign key connects two tables, therefore, the foreign key is mapped to a property. All attributes that are not foreign keys are also mapped to properties. All direct mapping systems follow these basic transformation patterns, as shown in Table 1.

3.2 Database normalization

The organization of database keys is central to both, core concepts in data modeling in databases, and, identification of ontological relationships from databases. In the former case this is known as *normalization*. For the latter, an example was presented in the Introduction and is the primary focus of Section 4. It is common for relational databases to be designed in Third Normal Form (3NF) and for direct mapping efforts to assume that the database is in 3NF. There are many database normal forms. The focus on 3NF is further justified mechanically as there are algorithms for translating databases into 3NF and data models in 3NF are considered best with respect to subjective assessment as to which form most frequently reflects intuition (Garcia-Molina *et al.*, 2009).

Different normalization schemes may yield different putative ontologies. In practice, relational databases will be normalized in different ways. The objective of normalization is to simplify the maintenance of data integrity and reduce the amount of space used for the database. However, schemas may be de-normalized for performance reasons. It is also natural to have multiple ontologies model the same domain. To illustrate the relevance to direct mapping, we consider an example of a database model for books and authors in three different normal forms.

If we were to create a relational model about books and their authors, a simple approach would be to consider a single table to hold all of this data, as shown in Table 2. However, the names of the authors are going to be repeated if we have several books with the same author. Therefore, we

could consider creating a table just for books and another table just for authors and creating a link between the book table and the author table.

The following examples show the resulting putative ontology if a database is in three different normal forms, 1NF, 2NF and 3NF. First, consider the following database in 1NF shown in Table 3.

There is a table for Authors and Books. The Books table has a foreign key that points to the Author table. The corresponding putative ontology that could be derived from a database in 1NF is shown in Figure 12.

The same way the database lacks higher normalization; a domain expert can consider that the ontology could be modeled in a better way. For example, the publisher should be a class and not a literal value. Now, consider the database in 2NF, as shown in Table 4.

The derived putative ontology is shown in Figure 13, which shows two new classes. These can be automatically generated if the database is in 2NF. However, if we further normalize the database to 3NF, as shown in Table 5 the derived putative ontology is the same ontology that is generated from the database in 2NF, because the table Authorship is transformed into a relationship between Author and Books.

This example does not prove that a putative ontology from a 2NF or 3NF database is the same. However, it does suggest that a putative ontology from a database may depend on the normalization.

A relational database can be represented in a logical model (UML, E-R model) or a physical schema (SQL-DDL). As a good software engineering practice, it is quite common to develop logical models for the relational schema, which then gets translated to a physical schema.

Table 3 Example of a database in 1NF

Author	
Author ID	Author name
1	John Smith
2	Jane Doe

Books					
ISBN	Title	Publisher	City	Subject	AuthorID
123-456	ABC	Publisher XYZ	Dallas	Science fiction	1
321-654	DEF	Publisher XYZ	Dallas	Science fiction	2
987-654	XYZ	Publisher ABC	Austin	Auto-biography	2

NF = Normal Form.

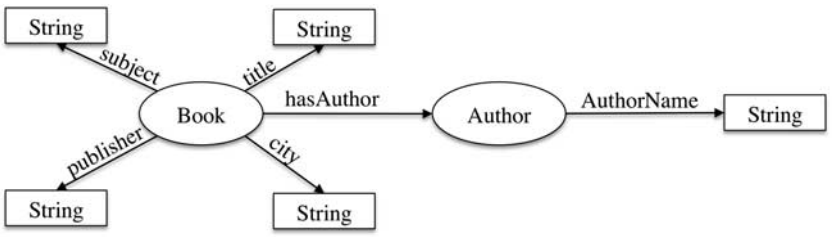

Figure 12 Ontology derived from database in 1NF from Table 3

Table 4 Example of a database in 2NF

Author ID	
Author ID	Author name
1	John Smith
2	Jane Doe

Books				
ISBN	Book Title	Publisher ID	Subject ID	Author ID
123-456	ABC	2	1	1
321-654	ABC	2	1	2
987-654	XYZ	1	2	2

Subject		
Subject ID	Subject Title	Type
1	Science fiction	Fiction
2	Auto-biography	Non-fiction

Publisher		
Publisher ID	Publisher name	City
1	Publisher ABC	Austin
2	Publisher XYZ	Dallas

NF = Normal Form.

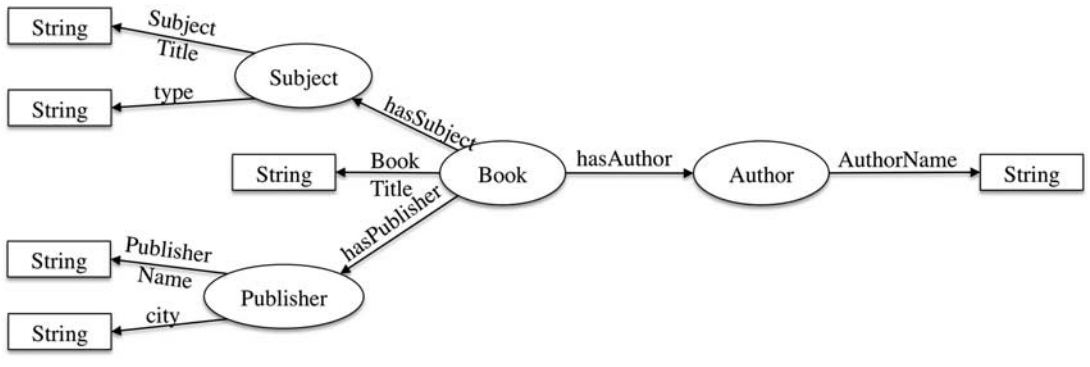

Figure 13 Ontology derived from database in 2NF from Table 4

Even after deployment, a database undergoes modifications due to changing application requirements. Such modifications are often not reflected in the logical model. Therefore, the physical schema, easily expressed in SQL-DDL, becomes the most accurate source for the structure of the database.

Table 5 Example of a database in 2NF

Author

Author ID	Author name
1	John Smith
2	Jane Doe

Books

ISBN	Title	Publisher ID	Subject ID
123-456	ABC	2	1
987-654	XYZ	1	2

Subject

SubjectID	Subject	Type
1	Science fiction	Fiction
2	Auto-biography	Non-Fiction

Publisher

PublisherID	Publisher	City
1	Publisher ABC	Austin
2	Publisher XYZ	Dallas

Authorship

ISBN	Author ID
123-456	1
321-654	2
987-654	2

NF = Normal Form.

Direct mapping approaches usually assume that the input database is in 3NF, even though they can still be applied if they are not in this normal form.

3.3 Inheritance modeling

Given that an SQL schema contains relationships between tables that can be expressed using only foreign keys, we find it necessary to identify foreign key patterns that can express only the

inheritance relationships. In other words, given a foreign key between two tables, is it possible to say that a subclass relationship exists between the tables involved? If such patterns exist, we can map them to subclass relationships in the ontology.

SQL-DDL does not provide a native mechanism to express inheritance. However, inheritance hierarchies can be modeled in a variety of ways in SQL schemas. Our university schema example shows two different modeling choices for inheritance. *STUDENT* and *PROFESSOR* tables are subclasses of *PERSON*, even though the relationships have been modeled differently. We present some inheritance modeling possibilities through possible foreign key patterns and discuss why some inheritance modeling choices are harder to identify automatically:

- *Foreign key is also the primary key*: An example of this case is the *PROFESSOR–PERSON* relationship in our university schema. This is a common pattern that identifies inheritance. An exception to this would be vertical partitioning of tables for performance reasons, as in some data warehousing applications. However, with our requirement of 3NF, such a scenario would not occur. Therefore, most direct mapping approaches automatically identify inheritance modeled in this way.
- *Foreign key and primary key are disjoint*: The *STUDENT–PERSON* relationship in our university schema is an example of this pattern. This pattern does not uniquely identify inheritance, and therefore cannot be automatically translated into an inheritance hierarchy in an ontology. A counterexample is the *COURSE–DEPT* relationship modeled in the same schema. In fact, this pattern is the most common one used for expressing one-to-many relationships.
- *Foreign key is a subset of the primary key*: This is another option for modeling inheritance in a relational database. However, other relationships can also be modeled this way, and therefore it is not a good candidate for automatic translation to an inheritance hierarchy in the ontology. A counterexample for this pattern is:

  ORDER(ONo)—ONo is the primary key;
  ORDERITEM(ONo,INo)—(ONo,INo) is the primary key, ONo is a foreign key to Order.

In a business domain, the relationship most likely means that an order item is a part of an order, instead of representing an inheritance relationship between the two entities.

3.4 Symmetric and transitive relationships

SQL-DDL lacks a native mechanism to represent symmetric and transitive relationships. However, these relationships can be expressed as triggers. Expressing such properties is natural in ontology languages like OWL.

The self-relation on the *OFFER* table, which represents a co-location of an offered course with another offered course, has interesting characteristics. First, it is symmetric, because if an offered course A is co-located with an offered course B, it means B is co-located with A as well. And second, it is transitive, which means that if A is co-located with B, and B is co-located with C, then A is co-located with C.

Such a self-relationship may or may not imply a symmetric or transitive relationship. Consider the example: *Employee(ID,Name,MgrID)*, where *ID* is the primary key, and *MgrID* is a foreign key to the *Employee* table itself that captures the manager's ID. Clearly, this relationship is not symmetric, because if John is the manager of Peter, that rules out the possibility of the same Peter being the manager of the same John.

Depending upon the domain, the relationship may or may not be transitive. If an employee's manager means any other employee higher in the organization, then being a manager is a transitive relationship. If it means only the immediate supervisor, then it is not a transitive relationship. The example shows that it is hard to identify logical characteristics of relationships in an SQL schema without using the domain knowledge.

3.5 Value constraints

An OWL value constraint puts constraints on the range of a property when applied to a particular class description. The three value constraints are *allValuesFrom*, *someValuesFrom* and *hasValue*. The OWL *allValuesFrom* is analogous to the universal quantifier of predicate logic while, the OWL *someValuesFrom* is analogous to the existential quantifier of predicate logic. Consider the following example:

Employee(id, id_role) Admin(id, id_role) Role(id_role,role_name)

Employee(id_role) and *Admin(id_role)* are foreign keys to *Role(id_role)*. One can consider this example as having two different relationships: Employee has a role and an Admin has a role. Therefore, two Object Properties would be created. Nevertheless, if both object properties result in meaning the same thing, then it would be sufficient to have only one Object Property and have an OWL *allValuesFrom* with the union of the Employee and Admin table. It is unclear on how to determine if these two Object Properties are the same.

Different value constraints may lead to different interpretations. For example, if we consider an OW assumption, then a property should only have a domain and not a range. If a property has a domain and a range, then we are considering a CW assumption. We could tighten the CW in a property by not only having a domain and range, but also having an OWL *allValuesFrom* constraint. In other words, while quantification can be expressed in ontologies, it is not natively represented in a relational database.

3.6 Check constraint

Check constraints are conditions that validate the data that is being inserted in a table. The enumerated *CHECK* constraint can be represented in OWL using OWL *oneof* and RDF *List*. For example, if we take attribute Session from the *SEMESTER* table in Figure 12; its representation in OWL would be the following:

oneof('Spring' 'Summer' 'Fall')

Nevertheless, the check constraint could represent inheritance. In this case, a *Spring Session* is also a type of *Session*. On the other hand, OWL is not expressive enough to represent all the possibilities that the semantics of *CHECK* offers. Consider having a *CHECK* constraint to guarantee that the value of a column degree is between 0 and 360. This type of constraint cannot be represented in OWL.

3.7 Open world vs. closed world

Relational databases usually operate under the CW assumption. This means that whatever is not in the database is considered false. The CW assumption is essential for important database concepts like integrity constraints and data validation (Drummond & Shearer, 2006). On the other hand, a knowledge-base community like the Semantic Web has an OW approach where whatever is not in the knowledge base is considered unknown. This assumption is natural for knowledge bases that often contain incomplete knowledge and grow with the manual discovery of new domain knowledge and automatic inferences.

Due to this difference, the concept of a constraint has very different meanings in the two worlds (Motik *et al.*, 2007). In a database setting, a constraint is mainly used for validation and prevents incorrect data from entering the database. In contrast, in an ontology, a constraint expresses some characteristics of classes or relationships but does not prevent assertion of any facts. Due to these constraints, some assertions may even result in unintuitive inferences.

Consider this example: In our university database, the relationship between a course and a department is expressed by a foreign key constraint. The foreign key constraint will allow the Course relation to have the record (CS386,CS,Databases) where CS is a department. However, the

record (CS386,John,Databases)—John is a student—will not be allowed because it violates the integrity constraint. In the ontology, the same constraint appears as object property *CODE*, with range restriction of *Dept* on the relationship. Unlike database constraints, the ontology constraint will not only allow the assertion of the triple *CODE(CS386,John)*, it will also infer that *John* is an instance of *Dept*, because of the range restriction on *CODE* property.

While there are obvious differences between CW and OW, OWs can be closed by explicitly stating required negations. OWL provides a way to state such facts by providing constructs to express disjoints.

When developing an ontology based on an SQL schema, it is very important to keep these differences in mind. The question of whether the OW should be closed or not, depends upon the domain and application requirements. If the ontology is for use within a particular application, it might make sense to close the world, whereas in a data integration setting, it might make more sense to have an OW.

4 Key combinations of a direct mapping transformation

We present all possible combinations of primary and foreign keys in an SQL-DDL to ontology transformations where the rules of the transformation system cover the entire range of possible relationships that can be described in an SQL schema. While it is trivial to translate relations into ontology classes, the existence of foreign keys represents relationships between corresponding classes, and poses a challenge for any transformation system. Multiple foreign keys may be present in a table, and each of them may be in a different form, representing different kinds of relationships, for example, one-to-one, one-to-many, etc. between the entities. The interaction of the foreign keys with primary keys provides clues to the properties of these relationships.

We have exhaustively considered all possible primary and foreign key combinations in order to infer semantic properties. This issue is complicated by the possible presence of compound keys (keys composed of multiple attributes) and the possibilities of multiple overlapping keys.

THEOREM 1 *The space of relations describable in SQL-DDL using various combinations of primary key and foreign key references between the relations can be partitioned into 10 disjoint cases of key combinations.*

The proof involves a syntactic enumeration of the cases and a closure operation over the space of relations. Figure 14 provides a useful summary of the theorem and its proof.

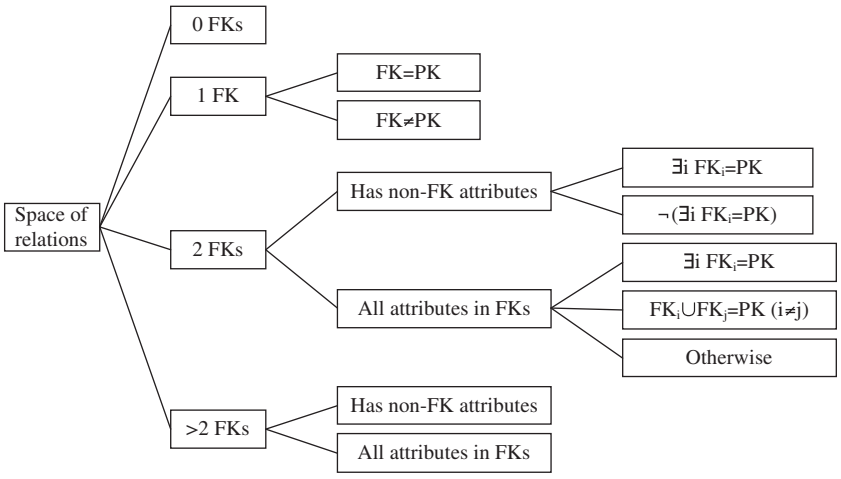

Figure 14 The tree describes the complete space of relations when all possible combinations of primary and foreign keys are considered

Proof. Briefly, we first partition the space by examining the number of foreign keys contained in relations. All relations without any foreign keys can be easily translated into classes in an ontology. Similarly, relations with more than two foreign keys usually represent n-ary relationships, and the rules for n-ary relationships are applicable. The cases for one or two foreign keys are more interesting and give rise to more possibilities including binary relations, inheritance and new classes. However, for each possible branch, we have carefully defined sets of rules for producing ontology classes and properties.

A table in a relational database can have either a Primary Key and/or Foreign Key. Both keys can include a number of attributes, x, where $x = 0 \ldots n$. There can only be one Primary Key in a table, whereas there can be one or more Foreign Keys. Therefore, our initial starting point is:

1. PK: a relation only has a Primary Key.
2. C-PK: a relation only has a composite Primary Key.
3. S-FK: a relation only has one Foreign Key.
4. N-FK: a relation has at least two or more Foreign Keys.

This can be represented in the following grammar:

$E \rightarrow PK + T|C\text{-}PK + T$
$E \rightarrow S\text{-}FK$
$E \rightarrow N\text{-}FK$
$T \rightarrow S\text{-}FK|N\text{-}FK$

By applying the closure operation for LR(0) item sets (Aho *et al*., 2006), the following additional cases are obtained, each corresponding to an item-set:

5. PK + S-FK: a relation has a Primary Key and only one Foreign Key
   (a) PK = S-FK: the Foreign Key is the Primary Key;
   (b) $PK \cap S\text{-}FK = 0$: the Foreign Key and the Primary Key do not share any attributes.
6. PK + N-FK: a relation has a Primary Key and two (2) Foreign Keys
   (a) $PK \cap N\text{-}FK = 0$: the Foreign Key and the Primary Key do not share any attributes;
   (b) $PK \subset N\text{-}FK$: one of the Foreign Keys is also the Primary Key.
7. PK + N-FK: a relation has a Primary Key and more than two (>2) Foreign Keys
   (c) $PK \cap N\text{-}FK = 0$: the Foreign Key and the Primary Key do not share any attributes;
   (d) $PK \subset N\text{-}FK$: one of the Foreign Keys is also the Primary Key.
8. C-PK + S-FK: a relation has a Composite Primary Key and only one Foreign Key.
   (a) $C\text{-}PK \cap S\text{-}FK = 0$: the Foreign Key and the Primary Key do not share any attributes;
   (b) $S\text{-}FK \subset C\text{-}PK$: the Foreign Key is part of the Primary Key.
9. C-PK + N-FK: a relation has a Composite Primary Key and two (2) Foreign Keys
   (a) $C\text{-}PK \cap N\text{-}FK = 0$: all the Foreign Keys and the Primary Key do not share any attributes;
   (b) $N\text{-}FK \subseteq C\text{-}PK$: all the Foreign Keys are part of the Primary Key;
   (c) $C\text{-}PK \cap N\text{-}FK \neq 0$, $C\text{-}PK - N\text{-}FK \neq 0$, $N\text{-}FK - C\text{-}PK \neq 0$: the Foreign Keys and Primary Key share common attributes.
10. C-PK + N-FK: a relation has a Composite Primary Key and more than two (>2) Foreign Keys
   (d) $C\text{-}PK \cap N\text{-}FK = 0$: all the Foreign Keys and the Primary Key do not share any attributes;
   (e) $N\text{-}FK \subseteq C\text{-}PK$: all the Foreign Keys are part of the Primary Key;
   (f) $C\text{-}PK \cap N\text{-}FK \neq 0$, $C\text{-}PK - N\text{-}FK \neq 0$, $N\text{-}FK - C\text{-}PK \neq 0$: the Foreign Keys and Primary Key share common attributes.

Therefore, a direct mapping transformation system should be complete with respect to all the cases of the theorem of key combinations.

5 Survey of direct mapping approaches

This section contains a review of seven different mapping efforts. The review is organized to compare these efforts based on their choice of target language, their handling of four aspects in the

source constructs and coverage of the 10 cases from Theorem 1. The four aspects are: translation of basic table structure, relationship between SQL attributes and constraints with respect to the ontology language, the handling of inheritance and n-ary relationships.

It is important to note that the relational model of a relational database is the basis to obtain RDF content. Relational data is to RDF as relational schema is to ontology (RDFS or OWL). Data is stored in a relational database that is defined with a schema. Likewise, RDF data is an instance of an ontology. After creating a putative ontology from the SQL schema, the data in the relational database can be extracted and transformed to ontological instances. All the following surveyed approaches accomplish this objective. Additionally, none of the surveyed approaches consider using triggers.

We acknowledge the overlap between the surveyed systems, denoted as the *general rule*, particularly when such overlap helped motivate the inclusion of a transform in the consolidated system. We offer counter-examples to any specific transformation where the syntactic underpinning of the source construct does not uniquely determine the ontological result. The analysis is summarized in Section 7.

5.1 Target language

The Semantic Web has two types of ontology languages: RDF Schema and OWL. It is important to denote a difference between them. In RDF Schema there is only one type of property while in OWL there are two: object properties and datatype properties. An object property is a relationship between two ontological classes. A datatype property is a relationship between an ontological class and a datatype (int, string, etc.). RDF Schema does not distinguish these two different types of properties; they are both treated the same.

Stojanovic *et al.* (2002a, 2002b) published the first approach that aimed at extracting domain semantics from SQL-DDL and transforming it to F-Logic and RDF Schema. Astrova's (2004) initial approach reverse engineers SQL-DDL and transforms it into an RDFS ontology. Buccella *et al.* (2004) published the first approach that takes SQL-DDL and transforms it to OWL through expository examples. Li *et al.* (2005) has the first approach that transform SQL-DDL to OWL with a combination of some formal notation and English language. Astrova *et al.* (2007) provides expository examples to describe a system for automatic transformation of a relational schema to an OWL ontology, which discovers more semantics than their previous effort (Astrova, 2004). However, the output from this system does not conform to OWL Descriptive Logic (OWL DL) restrictions. The work presented by Lubyte and Tessaris is an ontology extraction algorithm from a relational database logical schema (such as UML or ER) instead of a physical schema (SQL-DDL). Furthermore, the output is not in a Semantic Web language, instead it is a DLR-DB ontology language (Lubyte & Tessaris, 2009).

5.2 Source construct

We consider four source constructs, which are (i) translation of basic table structure, (ii) relationship between SQL attributes and constraints with respect to ontology language, (iii) the handling of inheritance and (iv) n-ary relationships. We will use the suggested correspondence between SQL and the Semantic Web as a basis for characterizing and comparing the different mapping efforts. The table construct corresponds to the Table Definition and RDFS layers of the stack in Figure 8. The attribute and constraints construct corresponds to the Constraint and OWL layers. Inheritance and n-ary relationships are modeling issues that each approach tackles differently. Therefore we see fit to compare how each approach addresses these two modeling issues.

5.2.1 Basic table structure

The *general rule* is:

- A many-to-many relationship represented as a table with exactly two attributes where the primary key is both attributes and each attribute is a foreign key is then mapped to an object

property that has a domain and range, which are the mapped ontological classes of the respective referenced tables.
- Otherwise, each table in a relational database is mapped to an ontological class.

Additionally, Astrova (2004) and Lubyte and Tessaris (2009) present each different type of table classifications. Astrova (2004) presents three different classifications for tables:

- *Base*: a table is independent of other tables.
- *Dependent*: the primary key of a table depends on another table, meaning that it is also a foreign key.
- *Composite*: a table that is neither base nor dependent.

The base and dependent tables are all mapped to ontological classes.

Lubyte and Tessaris (2009) presents four different classifications for tables:

- *Base*: the primary key and foreign key of a relation do not share attributes.
- *Specific*: a primary key is among the foreign key and either there is only one foreign key (the primary key and foreign key are the same attribute) or the table is referred to by another relation.
- *Relationship*: all the foreign keys compose the primary key.
- *Ambiguous*: all other tables.

The base and specific tables are mapped to ontological classes.

Stojanovic *et al*. (2002a), Astrova (2004) and Li *et al*. (2005) present an exception to the general rule, which is when several tables are used to describe a single entity. This is the case when vertical partitioning is considered in a relational database, where several tables share the same primary key. The result is a mapping of all the tables with the same primary key into a single ontological class. Astrova (2004) denotes this case *Key equality—Data equality—Attribute disjointedness*. Additionally, Astrova (2004) is the only approach to consider horizontal partitioning: when the key and attributes are equal across two different relations but the data is different (*Key and Attribute equality and Data disjointedness*). Both tables become a single ontological concept.

5.2.2 Attributes and constraints

For historic reasons, Stojanovic *et al*. (2002a, 2002b) and Astrova (2004) map to RDFS, that is, they published prior to the ratification of OWL. The *general rule* is that attributes in a table are mapped to a property, which has a domain that is the ontological class mapped from the respective table. An attribute that is both a foreign key and a primary key is ignored.

The other approaches map to OWL, therefore, there object properties and datatype properties are distinguished.

5.2.2.1 Web Ontology Language datatype properties. The *general rule* is that all non-foreign key attributes of a table are mapped to an OWL *Datatype Property*. The domain of the datatype property is the ontological class mapped from the respective table. The range of the datatype property is the respective datatype from the attribute.

Buccella *et al*. (2004) maps all attributes of a table to OWL *Functional Datatype Property*. If the attribute is NOT NULL, it is mapped to a cardinality constraint of 1. Primary Keys already have the *NOT NULL* implicit, therefore they are also mapped to a cardinality constraint of 1. However, Buccella *et al*. (2004) makes the assumption that datatype properties do not have domain and range defined because other classes can use them. Furthermore, the restriction OWL *allValuesFrom* is applied with a String. This approach would lead to ambiguity. Take the following example shown in Figure 15.

Following this rule, when a Datatype Property *name* has been created for the class *PERSON*, it would not have the domain *PERSON* and Range string, because another class could reuse it. The table *DEPT* also has an attribute name, therefore the previous Datatype Property created from the table *PERSON* could be reused for the new class *DEPT*. Because it is not explicit that the class

```
create table PERSON {
ID integer primary key,
NAME varchar not null }
```

```
create table DEPT {
  CODE varchar primary key,
  NAME varchar unique not null }
```

Figure 15 SQL-DDL of Person table and Dept table

PERSON and class *DEPT* are disjoint, it may be inferred that a name of a *DEPT* is also a name of a *PERSON*, which is false. Hence, due to the OW assumption, this rule is ambiguous.

Buccella *et al.* (2004) also consider *CHECK* constraint. For example, if a check constraint has *LIKE A%*, a class *beginning_with_A* can be created and that attribute can have a restriction that all values from that property have to be from that new class. Another possibility is to have the enumerated check constraints, which is then mapped to OWL *oneof*.

Li *et al.* (2005) generates cardinalities from the constraints of attributes in the relations. If an attribute is a primary key or foreign key then the minimum and maximum cardinality is 1. If an attribute is *NOT NULL*, the minimum cardinality is 1. If an attribute is *UNIQUE*, the maximum cardinality is 1.

Astrova *et al.* (2007) maps attributes with a maximum cardinality of 1 unless it is a foreign key. The property can also be defined as functional, which is the same as saying that the maximum cardinality is 1. The SQL data type is mapped to XML Schema Data types (XSD). If a *CHECK* constraint that verifies that an integer is greater than 0 is used, then the XSD used is positiveInteger. The attribute with a *UNIQUE* constraint is mapped to an OWL *Inverse Functional Property*. This rule cannot be true if the target ontology is OWL DL because Inverse Functional Properties can only be applied to Object Properties and not Datatype Properties (*OWL Guide 2004*). *UNIQUE* describes a key constraint, and would lead to computational intractability even in realistic ontologies, and are not supported by any implemented OWL reasoners (Lutz *et al.*, 2004). If the ontology is OWL Full, then it can be accepted. Attributes with a *NOT NULL* constraint are mapped to a minimum cardinality of 1. *PRIMARY KEY* constraints are mapped to OWL *Inverse Functional Property* (which encounters the same problem explained previously) and a minimum cardinality 1. Additionally *CHECK* constraints can be mapped several ways. If a column has a *CHECK* constraint to a single same value for all instances, then it is mapped to OWL *hasValue* restriction. Even though this translation makes sense, this rule appears to be ambiguous because it depends on a specific example, which is presented in the following example:

```
CREATE TABLE Project{
type VARCHAR CHECK (type ='Software')}
〈owl:hasValue rdf:datatype=''&xsd;string''〉Software〈/owl:hasValue〉
```

If the *CHECK* constraint is with enumeration, then it can be mapped to an enumerated datatype OWL *oneof*.

5.2.2.2 Object properties. The *general rule* is that an attribute that is a foreign key is mapped to an OWL *Object Property*. The domain of the object property is the ontological class mapped from the respective table and the range is the ontological class mapped from the reference table. Buccella *et al.* (2004) maps the foreign key to an OWL *Functional Object Property* with minimum cardinality of 1, where the domain is the mapped ontological class of the current table and the range is the mapped ontological class of the referenced table. The Inverse property is also created, which is not functional.

Buccella *et al.* (2004), Li *et al.* (2005) and Astrova *et al.* (2007) consider the case when the foreign key is a subset of the primary key. Buccella *et al.* (2004) maps the primary key attribute that is not a foreign key to an OWL *Functional Datatype Property* with cardinality of 1 and with the OWL *allValuesFrom* restriction, with the attributes datatype. The attribute that is a foreign key is mapped to an OWL *Functional Object Property*. This object property has a domain and range, where the mapped

ontological class of the current table is the domain, and the range is the mapped ontological class of the referenced table. The cardinality of 1 is also added and the inverse object property is also created. This rule is one of the inheritance modeling incompatibilities discussed in Section 3.3. On the other hand, Li *et al.* (2005) maps the foreign key to a *has-part* and *is-part-of* Object Property (Figure 6). For Astrova *et al.* (2007), if the foreign key is part of the primary key, it is mapped to an object property accompanied with its cardinality of 1.

Shen *et al.* (2006) adds the use of OWL *allValuesFrom* restriction. This is possible if a foreign key is mapped to an OWL *Object Property*, then the object property has an OWL *allValuesFrom* restriction, which refers to the corresponding inclusion dependency. This may lead to a disparity discussed previously in Section 3.5.

Astrova *et al.* (2007) consider scenarios that no other approach does: If a foreign key is a reference to its same table, then it is considered an OWL *Symmetric Property*. However, this rule cannot be always applied, which has been explained in Section 3.4. If a column in a table is a foreign key to the same table accompanied by *ON DELETE CASCADE*, the relationship consists of a whole part, where the part cannot exist without the whole; therefore it is an OWL *Transitive Property*.

5.2.3 Inheritance

There are different ways to model inheritance, even though there is not a standardized way of representing it in SQL-DDL, as discussed in Section 3.3. Therefore, there is no general rule to identify inheritance. We identify two patterns from the surveyed approaches:

SCENARIO 1. *Two tables share the same primary key and for one of the tables, the primary key is a foreign key*.

Both Lubyte & Tessaris (2009) and Astrova (2004) identify inheritance when a table has an attribute that is a primary key and a foreign key that references another table. Astrova (2004) denotes this as key equality and data inclusion. An example of the SQL-DDL for this scenario is shown in Figure 16 and the graphical representation of the ontological inheritance relationship is shown in Figure 17.

SCENARIO 2. *Two tables share the same primary key*.

For both Stojanovic *et al.* (2002a, 2002b) and Li *et al.* (2005), inheritance is an inclusion dependency between two relations and both relations are concepts. This rule conflicts when

```
create table STUDENT {
  STUDID integer primary key,
  NAME varchar}

create table PHDSTUDENT {
 STUDID  integer  primary  key
references STUDENT,
 YEAR date not null }
```

Figure 16 SQL-DDL of two tables that share the same primary key and for one of the tables, the primary key is the foreign key

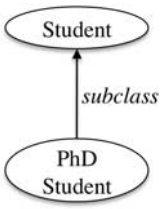

Figure 17 Ontological inheritance relationship

information is spread across several relations (vertical partitioning); in this case all the relations can be integrated into one concept. In this case, the user must decide which rule to apply, thus making this effort semi-automatic. Figure 18 shows the SQL-DDL that represents two tables sharing the same primary key, which could be mapped either to one same concept (vertical partitioning) or to an inheritance relationship.

Astrova (2004) expands this scenario by presenting three possible cases. The first case considers the case when two tables have key equality, data overlap and attribute disjointedness. In this case, a new ontological class is discovered, which has inheritance relationships from both the original tables. The SQL-DDL in Figure 16 would be translated to the ontology shown in Figure 19.

The second case considers key equality, data disjointedness and attribute overlap, as shown in the SQL-DDL in Figure 20. In this case, a new ontological concept is discovered, which would consist of the overlapping attributes. The two tables are now ontological classes that inherit from the new discovered class, as shown in Figure 21.

```
create table STUDENT {
  STUDID integer primary key,
  NAME varchar}

create table PHDSTUDENT {
 STUDID integer primary key,
 YEAR date not null }
```

Figure 18 SQL-DDL of two tables that share the same primary key

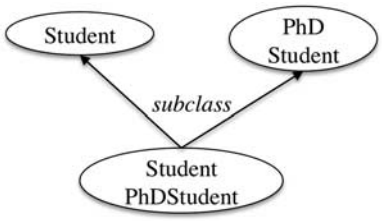

Figure 19 Inheritance structure from Astrova's (2004) key equality, data overlap and attribute disjointedness case

```
create table UNDERGRADSTUDENT {
  STUDID integer primary key,
  NAME varchar,
  YEAR varchar}

create table GRADSTUDENT {
 STUDID integer primary key,
 NAME varchar,
 RESEARCHAREA varchar }
```

Figure 20 SQL-DDL showing key equality, data disjointedness and attribute overlap

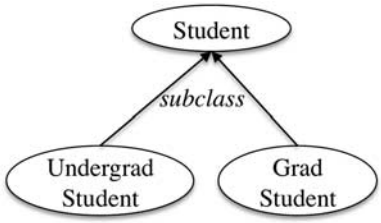

Figure 21 Inheritance structure from Astrova's (2004) key equality, data disjointedness and attribute overlap case

The third and final case considers key equality, data overlap and attribute overlap, where some data and attributes are common to both tables, then a 'diamond-shape' inheritance is generated where a class inherits from two super classes and the two super classes inherits from a common super class. An example of this resulting ontology is shown in Figure 22. Consider the case when a student is both an undergrad and a grad student. In this case it would be an instance of class *UndergradGradStudent*.

5.2.4 N-ary relationships

In Section 5.2, we described the general rule for a 2-ary relationship (table with exactly two attributes that are each foreign keys), which is mapped to an object property. This is the case when the table does not have any additional attributes. If it does have more than two attributes, then the *general rule* is to map the table to an ontological class. The attributes of the table follow the *general rule* of Section 5.3. Shen *et al*. (2006) presents a specific rule considering only 3-ary relationships that follows the previously described general rule. Figure 23 shows an example of a 3-ary relation and the result of applying the *general rule* is shown in Figure 24.

However, Li *et al*. (2005) is the only approach that differs from the general rule. If a table represents an n-ary relationship, then there is an object property that connects every single table with the other tables in the n-ary relationship. Taking this approach can produce unnecessary relationships. Consider the example of an n-ary relation from Figure 23. This rule would create the relationships Professor–Course, Professor–Semester and Course–Semester, as shown in Figure 25. Is the relationship Professor–Semester needed? Maybe the other relationships were already created and now they are repeated.

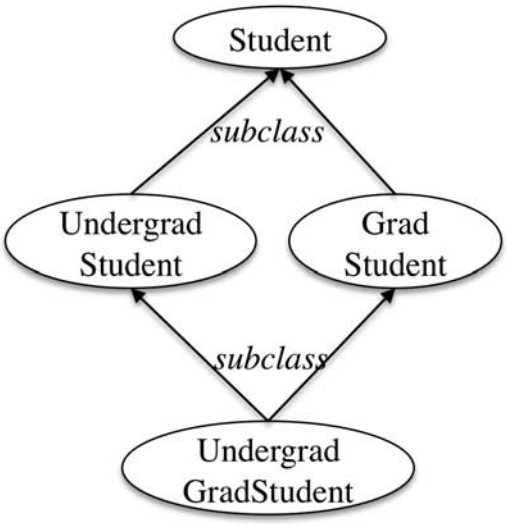

Figure 22 Inheritance structure from Astrova's (2004) key equality, data overlap and attribute overlap case

```
create table OFFER {
  ONO integer primary key,
  CNO integer foreign key
references COURSE(CNO),
  SNO integer foreign key
references SEMESTER(SNO),
  PID integer foreign key
references PROFESSOR(ID) }
```

Figure 23 SQL-DDL of the n-ary relationship Offer

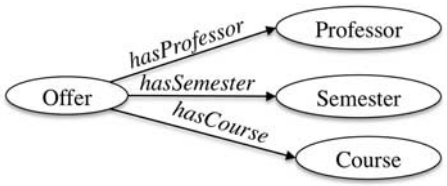

Figure 24 Ontological representation of an n-ary relationship Offer based on the general rule

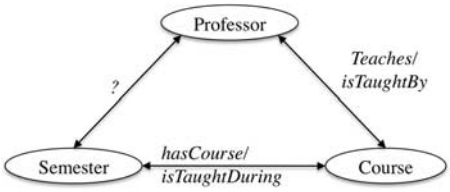

Figure 25 Ontological representation of the n-ary relationship Offer following Li *et al.*'s (2005) rule

5.3 Coverage of 10 cases of key combinations

In Theorem 1, we present 10 cases of primary and foreign key combinations. A direct mapping system should infer semantic properties by considering all these cases. In Table 6, we show the approaches that have formal rules and can be mapped to the different cases of Theorem 1. If an approach does not present formal rules, they do not appear in Table 9. Stojanovic *et al.* (2002a, 2002b) and Astrova (2004) were the first approaches to consider extracting the domain semantics from the SQL-DDL and transforming them into RDFS. The majority of their rules overlap, meaning that given the same SQL-DDL input, it would output the same RDFS. The difference between these two approaches is how they each identify inheritance. Stojanovic *et al.* (2002a, 2002b) presents the transformation rules in a formal manner. As shown in Table 6, Stojanovic *et al.*'s (2002a, 2002b) rules are not complete with respect to Theorem 1 in reference when a primary key is a subset of a foreign key. On the other hand, Astrova (2004) presents the transformations through examples without any formality, which may lead to ambiguity; for that reason, it cannot be determined if Astrova's (2004) approach is complete with respect to Theorem 1 and is not shown in Table 6.

Another series of surveyed direct mapping approaches transform the SQL-DDL to OWL ontologies. Li *et al.* (2005) does use a combination of formal notation and English language and has enumerated the rules, thus it is easier to identify which rules apply to specific cases. Table 6 shows that Li *et al.*'s (2005) approach has rules to satisfy each case of Theorem 1. Furthermore, Shen *et al.* (2006) extends on Li *et al.*'s (2005) approach, which is also presented in a combination of formal notation and English language. However, Shen *et al.* (2006) does not present rules to satisfy cases 7 and 10, as seen in Table 6, hence this approach is not complete with respect to Theorem 1. On the other hand, Buccella *et al.* (2004) and Astrova *et al.* (2007) identify more semantics from a relational databases schema as compared with Li *et al.* (2005). However, the output from this system does not conform to OWL DL restrictions. Additionally, they only present examples of transformation without any formalism at all, therefore the system is susceptible to ambiguities and there is no formal way to determine if they are complete with respect to Theorem 1, thus not showing up in Table 6.

Even though Lubyte and Tessaris (2009) also classifies relations based on the combination of primary keys and foreign keys, it is not possible to consider if this approach is complete with respect to all the cases of Theorem 1 because it may miss cases due to the fact that its input is a logical schema. For example, this approach does not consider the case if a relation has exactly two

Table 6 Rules that apply to each of the 10

Relation with	Stojanovic *et al.*[1]	Li *et al.*[2]	Shen *et al.*[3]	Consolidated system[4]
(1) PK	Rules C1, C2, C3	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 5
(2) C-PK	Rules C1, C2, C3,	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 5
(3) S-FK	Rules C1, C2, C3,	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(4) N-FK	Rules C1, C2, C3,	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(5) PK + S-FK				
(a) PK = S-FK	Rules C1, C2, C3, A3, I1	Rules 2, 7, 8, 9, 10, 11	Rules C-1, C-3, P-1.2, R-2, R-3, R-4	Rules 2, 5, 6
(b) PK ∩ S-FK = 0	Rules C1, C2, C3, A2	Rules 2, 3, 7, 9, 10, 11	Rules C-1, P-1.1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(6) PK + N-FK|N = 2				
(a) PK ∩ N-FK = 0	Rules C1, C2, C3, A1	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(b) PK ⊂ N-FK	Rules C1, C2, C3, A4	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5, 6
(7) PK + N-FK|N >2				
(a) PK ∩ N-FK = 0	Rules C1, C2, C3, A5	Rules 2, 7, 9, 10, 11	–	Rules 2, 4, 5
(b) PK ⊂ N-FK	–	Rules 2, 7, 9, 10, 11	–	Rules 2, 4, 5, 6
(8) C-PK + S-FK				
(a) C-PK ∩ S-FK = 0	Rules C1, C2, C3, A2	Rules 2, 3, 7, 9, 10, 11	Rules C-1, P-1.1, P-1.2, R-2, R-3, R-4	Rules 2, 5, 6
(b) S-FK ⊂ C-PK	–	Rules 2, 4, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(9) C-PK + N-FK|N = 2				
(a) C-PK ∩ N-FK = 0	Rules C1, C2, C3, A2	Rules 2, 3, 7, 9, 10, 11	Rules C-1, P-1.1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(b) N-FK ⊆ C-PK	Rules C1, C2, C3, A4	Rules 2, 5, 6, 7, 9, 10, 11	Rules C-1, P-2, P-1.2, R-2, R-3, R-4	Rules 2, 3, 4, 5
(c) C-PK ∩ N-FK ≠ 0, C-PK − N-FK ≠ 0, N-FK − C-PK ≠ 0	–	Rules 2, 7, 9, 10, 11	Rules C-1, P-1.2, R-2, R-3, R-4	Rules 2, 4, 5
(10) C-PK + N-FK|N >2				
(a) C-PK ∩ N-FK = 0	Rules C1, C2, C3, A2	Rules 2, 3, 7, 9, 10, 11	–	Rules 2, 4, 5
(b) N-FK ⊆ C-PK	Rules C1, C2, C3, A4	Rules 2, 5, 6, 7, 9, 10, 11	–	Rules 2, 3, 4, 5
(c) C-PK ∩ N-FK ≠ 0, C-PK − N-FK ≠ 0, N-FK − C-PK ≠ 0	–	Rules 2, 7, 9, 10, 11	–	Rules 2, 4, 5

[1]The rule numbers correspond to the same rule numbers of Stojanovic *et al.* (2002a, 2002b).
[2]The rule numbers correspond to the same rule numbers of Li *et al.* (2005).
[3]The rule numbers correspond to the same rule numbers of Shen *et al.* (2006).
[4]The rule numbers correspond to the same rule numbers of Section 6.

foreign keys and they are both primary keys. In this case, the relation is mapped to an association itself between two classes, because this relation denotes a relationship between two other relations. In a logical schema, this binary relation would not exist because it would be represented as a direct link between the two relations. However, when translating the logical schema to the physical schema, this relation appears. Due to the fact that this work does not target a Semantic Web language, it is not possible for it to be completely evaluated with respect to all the other previously surveyed approaches.

Finally, in Table 5, we also present the rules of the consolidated system that satisfy all of the 10 cases. The consolidated system is explained in Section 6.

6 A consolidated system of direct mapping of a relational database to the Semantic Web

Considering the contributions of the methods above, we derive a consolidated system for automatic transformation of relational databases into OWL ontologies. The defined transformations are in first-order logic (FOL; Tirmizi *et al.*, 2008).

6.1 Assumptions

We make the following assumptions:

- *The relational schema, in its most accurate form, is available in SQL-DDL.* As a good software engineering practice, it is quite common to develop logical models for the relational schema. However, even after deployment, a database undergoes modifications due to changing application requirements. Such modifications are often not reflected in the logical models. Therefore, the physical model, easily expressed in SQL-DDL, becomes 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 (Du & Wery, 1999; Wang *et al.*, 2000).

6.2 Predicates and functions

We define 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. The examples relate to Figure 10:

Rel(r)	r is a relation (or table) identified by CREATE TABLE statement
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); for example: *NN(NAME,PERSON)* holds
Unq(x,r)	*x* is an attribute (or a set of attributes) in relation *r* with UNIQUE constraint; for example *Unq({NAME},DEPT)* holds
Chk(x,r)	x is an attribute in relation *r* with enumerated list (CHECK IN) constraint; for example *Chk(SESSION,SEMESTER)* holds
PK(x,r)	x is the (single or composite) primary key of relation *r* identified using the PRIMARY KEY constraint; for example: *PK({OFFER,SID},STUDY)* holds; also: $PK(x,r) \rightarrow Unq(x,r) \wedge NN(x,r)$
FK(x,r,y,s)	x is a (single or composite) foreign key in relation *r* and references *y* in relation *s*; for example: *FK({ID},STUDENT,{ID},PERSON)* is satisfied
NonFK(x,r)	x is an attribute in relation *r* that does not participate in any foreign key; for example: *NonFK(NAME,DEPT)* holds, *NonFK(SID,REG)* does not hold

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 a datatype 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, that is, inverse of a functional property
Crd(p,m,v)	the (maximum and minimum) cardinality of property *p* for class *m* is *v*
MinC(p,m,v)	the minimum cardinality of property *p* for class *m* is *v*
MaxC(p,m,v)	the maximum 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 rules in the remainder 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 *x*; undefined if there is no such foreign key
type(x)	maps an attribute *x* to its suitable OWL recommended data type (data types are discussed in more detail in a later section)
list(x)	maps an attribute *x* to the list of allowed values for it; this function is applicable only to attributes that have a CHECK IN constraint defined on them, that is, *Chk(x,r)* 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 define *BinRel* as follows:

Rule Set 1:

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

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.

6.3 Transformation rules and examples

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

6.3.1 Producing unique identifiers (URIs) and label

The concept of globally unique identifiers is fundamental to OWL ontologies, thus it is important to understand how we can produce identifiers and names for classes and properties that form the ontology.

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.

For the purposes of this paper, we have not used fully qualified names in our examples. When needed, we append a name with an integer to make it unique, for example, ID1, ID2, etc.

6.3.2 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

Table 7 Some common SQL data types and corresponding XML Schema types recommended for OWL

SQL data type	XML schema type	SQL data type	XML schema type
INTEGER	xsd:integer	VARCHAR	xsd:string
DECIMAL	xsd:decimal	CHAR	xsd:string
FLOAT	xsd:float	DATE	xsd:date
REAL	xsd:double	TIME	xsd:time
BOOLEAN	xsd:boolean	TIMESTAMP	xsd:dateTime

XML = extensible markup language; OWL = Web Ontology Language.

a subset of XML Schema types (Biron *et al.*, 2004) in Semantic Web ontologies (Dean & Schreiber, 2004; Hayes, 2004).

In Table 7 we present a list of commonly used SQL data types along with their corresponding XML Schema types. During transformation of datatype properties, the SQL data types are transformed into the corresponding XML Schema types.

6.3.3 Identifying classes

According to the OWL Language Guide (Smith *et al.*, 2004), 'the most basic concepts in a domain should correspond to classes …'. Therefore we would expect basic entities in the data model to translate into classes in an OWL ontology.

Given the definition of a binary relation, it is 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 Rule 2.

Rule Set 2:

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

In combination with Rule Set 1, a binary relation has exactly two foreign keys and no other attributes, 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. In our example schema, the *PERSON* table does not have a foreign key, so *Rel(PERSON)* is true and *BinRel(PERSON,_,_)* is false. Therefore, we conclude *Class(PERSON)*, that is, *Person* should be mapped to a class. The same reasoning holds for the *Dept* and *Semester* tables.
- All tables that have one foreign key should also be transformed to classes. No such tables can satisfy the *BinRel* predicate. Using the same rule we conclude that *STUDENT*, *PROFESSOR* and *COURSE* should be mapped to classes.
- The tables with more than two foreign keys should be transformed to classes as well. Such tables may represent an entity (when they have attributes not appearing in a foreign key) or an n-ary relationship between entities (where all attributes appear in foreign keys). Fortunately, in OWL, both these cases can be modeled the same way, that is, by translating the entity or the n-ary relationship into a class (Noy & Rector, 2006). From our running example, *OFFER* represents an n-ary relationship, and can be modeled as a class using the given rule.
- For tables containing exactly two foreign keys, the presence of independent attributes qualifies them to be treated as entities instead of binary relations, and translated to classes in OWL ontologies. The table *STUDY*, with an independent attribute *Grade*, is an example of this case, and is translated to an OWL class.

As a result of applying Rule Set 2, we have identified the classes (see Table 8) from our relational schema.

6.3.4 Identifying object properties

A property is a relationship that lets us assert general facts about the members of classes. There are two major types of properties, object properties and datatype properties (Smith *et al.*, 2004).

Table 8 Classes identified from the relational schema by applying Rule Set 2

Classes			
Class(PERSON)	*Class(STUDENT)*	*Class(PROFESSOR)*	*Class(DEPT)*
Class(SEMESTER)	*Class(COURSE)*	*Class(STUDY)*	*Class(OFFER)*

Properties have a domain, which defines the subject, and a range, which defines the object (Dean & Schreiber, 2004). We will describe datatype properties in the next section.

An object property is a relation between instances of two classes in a particular direction. The direction is from the domain of the object property to the range of the object property. In practice, it is often useful to define the direction of an object property in both directions, creating a pair of object properties that are inverses of each other. OWL provides 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, written r' in our notation, such that, *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:

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

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, only the *Reg* table fits the condition. *Reg* is a binary relation between *STUDENT* and *SEMESTER* entities, which are not binary relations. Therefore, *ObjP(REG,STUDENT,SEMESTER)* holds, and since we can create inverses, *ObjP(REG′,SEMESTER,STUDENT)* and *Inv(REG,REG′)* 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 (datatype or object) property with minimum cardinality of 0 and maximum cardinality of 1 is called a *functional property*, represented as *FP* in our rules. If an object property is functional, then its inverse is an *inverse functional property*, represented as *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 illustrate the use of these predicates.

Rule Set 4 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*—representing foreign keys not in or referencing binary relations—and then express the rules in terms of this predicate.

Rule Set 4:

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

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

Table 9 Some object properties identified from the relational schema by applying Rule Sets 3 and 4. An object property P implies the existence of an inverse property P'. Due to lack of space, we explicitly specify the inverse property only for the first property

Object Properties
ObjP(REG,STUDENT,SEMESTER), ObjP(REG′,SEMESTER,STUDENT), Inv(REG,REG′)
ObjP(ID1,STUDENT,PERSON), FP(ID1), FP(ID1′), Crd(ID1,STUDENT,1)
ObjP(CODE,COURSE,DEPT), FP(CODE), IFP(CODE′), Crd(CODE,COURSE,1), MinC(CODE′,DEPT,0)
ObjP(CNO,OFFER,COURSE), FP(CNO), IFP(CNO′), MinC(CNO′,COURSE,0)
ObjP(CONO,OFFER,OFFER), FP(CONO), IFP(CONO′), MinC(CONO′,OFFER,0)
ObjP(RNO,STUDY,STUDENT), FP(RNO), IFP(RNO′), MinC(RNO′,STUDENT,0)

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 can reference at most one record (instance) of the range, the object property is functional. This requires that inverse of that object property is inverse functional. One example from our university schema is the foreign key from *Study* to *Student* which gives us *ObjP(RNO,STUDY,STUDENT)*, *FP(RNO)*, *Inv(RNO′,RNO)*, *ObjP(RNO′,STUDENT,STUDY)* and *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 already covered by functional property predicate. With this constraint (rule 4b), we can set the maximum and minimum cardinality to 1.
- These rules imply, a minimum cardinality restriction of 0 on the inverse property. An instance in the range can be referenced by any number of instances in the domain, thus, there is no 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 the NOT NULL constraint. Like one-to-many relationships mentioned above, if NOT NULL is present, it 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.

We do not create an object property if the foreign is being equal to the primary key. Instead, we consider it as a pattern for inheritance and propose a rule for inheritance mapping. On the other hand, we are unable to capture the inheritance between *STUDENT* and *PERSON*, as it is not a unique inheritance pattern, and we transform this relationship into an object property.

A list showing some object properties and their characteristics obtained from the sample relational schema by applying Rule Sets 3 and 4 are presented in Table 9.

6.3.5 Identifying datatype properties

Datatype properties are relationship between instances of classes with RDF literals and XSD. Like object properties, datatype 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.

Table 10 Some datatype properties identified from the relational schema by applying Rule Set 5

Datatype properties
DTP(ID1,PERSON,xsd:integer), FP(ID1), Crd(ID1,PERSON,1)
DTP(NAME1,PERSON,xsd:string), FP(NAME1), Crd(NAME1,PERSON,1)
DTP(ROLLNO,STUDENT,xsd:integer), FP(ROLLNO), Crd(ROLLNO,STUDENT,1)
DTP(DEGREE,STUDENT,xsd:string), FP(DEGREE)
DTP(SNO,SEMESTER,xsd:integer), FP(SNO), Crd(SNO,SEMESTER,1)
DTP(YEAR,SEMESTER,xsd:date), FP(YEAR), Crd(YEAR,SEMESTER,1)
DTP(SESSION,SEMESTER,xsd:string ∩ {SPRING,SUMMER,FALL}), FP(SESSION)
DTP(GRADE,STUDY,xsd:string), FP(GRADE)

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

Rule Set 5:

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

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

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

Rule Set 5 says that attributes that do not contribute towards foreign keys can be mapped to datatype properties with range equal to their mapped OWL type. Because each record can have at most one value per attribute, each datatype 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 datatype properties extracted from our sample university database schema are listed in Table 10.

6.3.6 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 (Smith *et al.*, 2004).

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

The following rule describes a common case that can be used 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*. In our sample university schema, we can clearly identify that *Subclass(PROFESSOR,PERSON)* holds.

6.4 Coverage of 10 case of key combinations

We have defined the entire set of transformation rules in first-order logic eliminating the possibility of syntactic and semantic ambiguities and complete with respect to Theorem 1. The use of first-order logic also allows for easy implementation of the system in rule-based languages. In defining the consolidated rules, we have ensured compatibility with description logics based OWL

sublanguage (OWL DL), which is essential to assuring decidability for reasoning. In Table 5, we show how the rules of the consolidated system satisfy all 10 cases.

6.5 Implementation

As a result of applying our rules on the given relational schema, we derive the ontology shown in Table 9.

A comparison of the ontologies produced by the domain expert (Figure 11) with the one produced automatically using our rules (shown in Figure 26) shows a number of differences. For example, the rules of the consolidated system are unable to capture the subclass relationship of *STUDENT* with *PERSON*, or the symmetric and transitive characteristics of the co-location relationship among *OFFER* instances. These examples clearly show that automatic translation of a relational schema to an ontology has some limitations, and that these limitations are inline with the disparities we have identified earlier.

As a result of presenting this consolidated system in FOL, these rules have been implemented in the JESS rule language; therefore the FOL expression is stratified.

For the sake of this example, we have implemented the rules in JESS. To obtain the SQL-DDL, we analyze only the data dictionary of MySQL databases. The ontology is then generated using the Jena framework.

Automatically Produced Ontology

```
Ontology(<urn:sql2owl>

 ObjectProperty(<REG> domain(<STUDENT>) range(<SEMESTER>))
 ObjectProperty(<REG_I> inverseOf(<REG>))
 ObjectProperty(<COURSE.DEPTCODE> Functional domain(<COURSE>) range(<DEPT>))
 ObjectProperty(<COURSE.DEPTCODE_I> InverseFunctional inverseOf(<COURSE.DEPTCODE>))
 ObjectProperty(<OFFER.CONO> Functional domain(<OFFER>) range(<OFFER>))
 ObjectProperty(<OFFER.CONO_I> InverseFunctional inverseOf(<OFFER.CONO>))
 ObjectProperty(<STUDENT.ID> Functional InverseFunctional
  domain(<STUDENT>) range(<PERSON>))
 ObjectProperty(<STUDENT.ID_I> Functional InverseFunctional
  inverseOf(<STUDENT.ID>))

 ...

 DatatypeProperty(<COURSE.CNO> Functional domain(<COURSE>) range(xsd:integer))
 DatatypeProperty(<SEMESTER.YEAR> Functional domain(<SEMESTER>) range(xsd:date))
 DatatypeProperty(<SEMESTER.SESSION> Functional domain(<SEMESTER>)
  range(oneOf("SPRING" "SUMMER" "FALL")) range(xsd:string))
 ...

 Class(<PERSON> partial ...)
 Class(<PROFESSOR> partial <PERSON> ...)
 Class(<STUDENT> partial restriction(<STUDENT.ID> cardinality(1))
  restriction(<STUDY.RNO_I> minCardinality(0)) ...)
 Class(<COURSE> partial restriction(<COURSE.DEPTCODE> cardinality(1))
  restriction(<COURSE.CNO> cardinality(1)) ...)
 ...
)
```

Figure 26 Parts of an ontology corresponding to the University Database, produced automatically by applying the rules on the given SQL-DDL. The output format is Web Ontology Language Abstract Syntax

7 Analysis and conclusion

We believe that the following are fundamental ingredients for a transformation system aiming for the migration of relational data to the Semantic Web:

- The rules for a transformation system should be formally specified to avoid any syntactic or semantic ambiguities in the specifications. Rules defined in formal systems like first-order logic can be implemented in languages such as Prolog, Datalog, JESS or Drools.
- When developing rules for automatic translation of a relational database to an ontology, special care should be taken to avoid the influence of domain-specific examples, thus an automated transformation system should try to capture the semantics offered by SQL-DDL alone. Sometimes, the influence of examples from a particular domain can result in incorrect rules that are based on enumerating examples instead of focusing on formal power.
- The transformation system should infer semantic properties by considering all the possible combination of foreign and primary keys, hence satisfying all the cases of Theorem 1.

In this work, we have reviewed seven different approaches that automatically expose relational databases to the Semantic Web. We introduce the concept of the *putative ontology*, which is the resulting ontology of direct mapping an SQL schema and exposes the relational data as RDF instances of the putative ontology.

We expose patterns between relational databases and ontologies, and the difficulties when it comes to identifying richer semantics in a relational database schema. First, we show how normalization can affect the putative ontology and demonstrate that inheritance in a relational database schema can be modeled in different ways; therefore a single rule to identify inheritance is not possible. In addition, symmetric and transitive relationships may be implicit in the domain but are not explicit in the relational schema. Furthermore, identifying specific value constraints such as *for all* or *there exist* may imply identifying new properties or reusing existing properties. Continuously, it may also have implications in the OW and CW assumptions. Additionally, SQL's check constraint can represent different types of semantics such as enumeration, inheritance or business-type rules. Finally, we recognize that analyzing triggers to extract semantics applicable to ontologies is still an open problem. Overall, there is also a need to evaluate the quality of a putative ontology and a domain ontology.

We present Theorem 1 that states that a direct mapping transformation must consider all the possible primary key and foreign key combinations. We compare all the approaches based on four source constructs (Table, Attributes and constraints, Inheritance and n-ary relationships) and identify which approaches satisfy Theorem 1. Finally, we present a consolidated direct mapping system.

In Table 11, we summarize the output of each surveyed approach based on the four source constructs: basic table structure, attributes and constraints, inheritance and n-ary relationships. In Table 12, we present the output of each direct mapping transformation system given the SQL-DDL input. Some cells may be empty because the system does not offer any transformation from the given SQL construct. The work of Lubyte and Tessaris (2009) is not shown in Table 12 because their input is not SQL-DDL and their output is not in RDFS or OWL. However, their work is proven to be an equivalence preserving schema transformation.

There is a major overlap in the output of all direct mapping systems. Stojanovic *et al.* (2002a, 2002b) and Astrova (2004) only output RDFS ontologies. Buccella *et al.* (2004) and Shen *et al.* (2006) are the only approaches that considers using OWL *allValuesFrom*, however they are used differently. This leads us to acknowledge that identifying the use of OWL *allValuesFrom* and OWL *someValuesFrom* directly from a relational schema is still an open problem, as expressed in Section 3.5. In addition, we have identified different uses of cardinalities. Some approaches present the use of OWL *maxCardinality* and OWL *minCardinality* while others use only OWL *cardinality*.

In Table 13, we present the specific RDFS and OWL constructs that are created through each direct mapping approach. We note that Stojanovic *et al.* (2002a, 2002b) and Astrova (2004) output the same RDFS constructs. Li *et al.* (2005) and Shen *et al.* (2006) also produce the same OWL outputs, for the exception that Shen *et al.* (2006) is able to use OWL *allValuesFrom*. Astrova *et al.* (2007)

Table 11 Output of each direct mapping approach given a Source Input

	Basic table structure	Attribute and constraints	Inheritance	N-ary relationship
General Rule	• Many-to-many relationship is mapped to a RDF *property* or OWL *Object Property* • Otherwise table is mapped to an ontological class	RDFS: • Table attributes are mapped to a RDF *property* OWL: • Non-foreign key attributes mapped to an OWL *Datatype Property* • Foreign key attributes are mapped to an OWL *Object Property*	No general rule.	• Map the table that represents a n-ary relationship to an ontological class
Stojanovic *et al.*	• Follows general rule • Vertical partitioned tables mapped to a single ontological class	• Follows RDFS general rule	• *Two tables share the same primary key*	• Follows general rule
Astrova (2004)	• Follows general rule • 3 classifications: base, dependent and composite • Vertical partitioned tables mapped to a single ontological class • Horizontal partitioned tables mapped to the a single ontological class	• Follows RDFS general rule	• Two tables share the same primary key and for one of the tables, the primary key is a foreign key • Two tables have key equality, data overlap and attribute disjointedness • Key equality, data disjointedness and attribute overlap between two tables • Key equality, data overlap and attribute overlap between two tables	• Follows general rule
Buccella *et al.* (2004)	• Follows general rule	• Follows OWL general rule • Maps all attributes to OWL *Functional Datatype Property*		• Follows general rule

Table 11 *(Continued)*

		• PRIMARY KEY and NOT NULL attributes mapped to cardinality of 1 • Domain and Range not considered • OWL *allValuesFrom* restriction is applied with a String • Foreign Key attributes mapped to OWL *Functional Object Property* with min cardinality of 1		
Li *et al*. (2005)	• Follows general rule • Vertical partitioned tables mapped to a single ontological class	• Follows OWL general rule • PRIMARY KEY and FOREIGN KEY attributes have a min and max cardinality of 1 • NOT NULL attributes have min cardinality of 1 • UNIQUE attributes have max cardinality of 1	• *Two tables share the same primary key*	• OWL object property created that connects every single table with the other tables in the n-ary relationship
Shen *et al*. (2006)	• Follows general rule	• Follows OWL general rule • Foreign Key attribute has OWL *allValuesFrom* restriction	• *Two tables share the same primary key*	• Follows general rule
Astrova *et al*. (2007)	• Follows general rule • Vertical partitioning: maps all tables with same primary key to single ontological class	• Follows OWL general rule • All non foreign key attributes have max cardinality of 1 • Attribute with CHECK constraint with int greater than 0 mapped to xsd:positiveInteger • UNIQUE attribute mapped to OWL *Inverse Functional Property*		• Follows general rule

Table 11 *(Continued)*

	Basic table structure	Attribute and constraints	Inheritance	N-ary relationship
		• *PRIMARY KEY* attribute mapped to OWL *Inverse Functional Property* with min cardinality of 1 • CHECK constraint mapped to OWL *hasValue* and *oneof* • *FOREIGN KEY* references itself then mapped to OWL *Symmetric Property* • *FOREIGN KEY* as *ON DELETE CASCADE* restriction then mapped to OWL *Transitive Property*		
Lubyte and Tessaris (2009)	• Follows general rule • 4 classifications: base, specific, relationship, ambiguous	• Table attributes are mapped to a property	• Two tables share the same primary key and for one of the tables, the primary key is a foreign key	• Follows general rule

RDF = Resource Description Framework; RDFS = Resource Description Framework Schema; OWL = Web Ontology Language.

Table 12 Ontological output of each direct mapping approach given the SQL input

	Stojanovic *et al.* (2002a, 2002b)	Astrova (2004)	Buccella *et al.* (2004)	Li *et al.* (2005)	Shen *et al.* (2006)	Astrova *et al.* (2007)	Consolidated System
Target Ontology Language	RDFS/F-Logic	RDFS/F-Logic	OWL	OWL	OWL	OWL Full	OWL DL
⟨table name⟩	RDFS Class RDFS Subclass RDFS Property RDFS Domain RDFS Range	RDFS Class RDFS Subclass RDFS Property RDFS Domain RDFS Range	OWL Class RDFS Subclass OWL ObjectProperty RDFS Domain RDFS Range	OWL Class RDFS Subclass OWL ObjectProperty RDFS Domain RDFS Range	OWL Class RDFS Subclass OWL ObjectProperty RDFS Domain RDFS Range	OWL Class RDFS Subclass OWL ObjectProperty RDFS Domain RDFS Range	OWL Class RDFS Subclass OWL ObjectProperty RDFS Domain RDFS Range
⟨column name⟩	RDFS Property RDFS Domain	RDFS Property RDFS Domain	OWL Datatype Property OWL Functional Property OWL *allValuesFrom*	OWL Datatype Property OWL Object Property RDFS Domain	OWL Datatype Property OWL Object Property RDFS Domain	OWL Datatype Property OWL Object Property RDFS Domain	OWL Datatype Property OWL Object Property OWL Functional Property RDFS Domain
⟨data type⟩	RDFS Range	RDFS Range	OWL Functional Property OWL AllValuesFrom XML Schema Datatypes	RDFS range	RDFS range	OWL Maximum Cardinality of 1 RDFS range	OWL Maximum Cardinality of 1 OWL Functional Property RDFS range XML Schema Datatypes
⟨referential constraint definition⟩	RDFS Property	RDFS Property	OWL Object Property OWL Functional Property OWL InverseOf	OWL Object Property OWL Minimum Cardinality of 1 OWL Maximum Cardinality of 1	OWL Object Property OWL Minimum Cardinality of 1 OWL Maximum Cardinality of 1 OWL AllValuesFrom	OWL Object Property OWL Cardinality of 1 OWL Symmetric Property OWL Transitive Property	OWL Object Property OWL Functional Property OWL InverseOf OWL Minimum Cardinality of 0
⟨primary key definition⟩	F-Logic axioms	F-Logic axioms	OWL Cardinality of 1	OWL Minimum Cardinality of 1 OWL Maximum Cardinality of 1	OWL Minimum Cardinality of 1 OWL Maximum Cardinality of 1	OWL Inverse Functional Property OWL Minimum Cardinality of 1	OWL Cardinality of 1 OWL Functional Property
⟨unique constraint definition⟩	F-Logic axioms	F-Logic axioms		OWL Maximum Cardinality of 1	OWL Maximum Cardinality of 1	OWL Inverse Functional Property	OWL Inverse Functional Property
⟨not null constraint definition⟩	F-Logic axioms	F-Logic axioms	OWL Cardinality of 1	OWL Minimum Cardinality of 1	OWL Minimum Cardinality of 1	OWL Minimum Cardinality of 1	OWL Cardinality of 1
⟨check constraint definition⟩						OWL Datatype Property OWL hasValue OWL oneOf	OWL Datatype Property OWL oneOf

RDFS = Resource Description Framework Schema; OWL = Web Ontology Language; OWL DL = OWL Description Logic; XML = extensible markup language.

Table 13 RDFS/OWL constructs that are outputted by each direct mapping approach

	Stojanovic *et al.* (2002a, 2002b)	Astrova (2004)	Buccella *et al.* (2004)	Li *et al.* (2005)	Shen *et al.* (2006)	Astrova *et al.* (2007)	Consolidated System
owl:class			x	x	x	x	x
rdfs:class	x	x					
owl:allValuesFrom			x		x		
owl:someValuesFrom							
owl:hasValue						x	
owl:maxCardinality			x	x	x	x	x
owl:minCardinality			x	x	x	x	x
owl:cardinality			x				x
owl:intersectionOf							
owl:unionOf				x	x		
owl:complementOf							
rdfs:subClassOf	x	x		x	x	x	x
owl:equivalentClass							
owl:disjointWith							
owl:oneOf						x	x
owl:DatatypeProperty			x	x	x	x	x
owl:ObjectProperty			x	x	x	x	x
rdfs:subPropertyOf							
rdfs:domain	x	x	x	x	x	x	x
rdfs:range	x	x	x	x	x	x	x
owl:equivalentProperty							
owl:inverseOf			x	x	x	x	x
owl:FunctionalProperty			x				x
owl:InverseFunctionalProperty						x	x
owl:TransitiveProperty						x	
owl:SymmetricProperty						x	

RDFS = Resource Description Framework Schema; OWL = Web Ontology Language.

is the only approach able to output OWL *SymmetricProperty*, OWL *TransistiveProperty* and OWL *hasValue*. However, the rules to output these specific constructs are ambiguous and therefore not recommendable. We acknowledge that identifying symmetric and transitive properties from a relational schema is also an open problem.

8 Future directions

The Semantic Web is showing a constant growth pattern. This has been attributed to the Linked Open Data project[1] created by the W3C's Semantic Web Education and Outreach Interest Group[2]. The main goal of this project is to publish data sets as RDF on the web and create links between them. For common web users, the Semantic Web does not yet have great significance; therefore it is rare that integrating a relational database with the Semantic Web accrues obvious benefit to the owners of a website. It is expected that the Semantic Web will become mainstream and more accessible to common web users; therefore the need to integrate databases to the Semantic Web will increase. The W3C foresees this issue and has already organized the Relational Database to RDF (RDB2RDF) working group[3] in order to recommend a standardized mapping language between relational databases to RDF (Das *et al*., 2010) and a standard way of direct mapping a relational database to RDF (Arenas *et al*., 2010). The work presented in this survey has highly influenced this standard and test cases.

Although there are many proposals for direct mapping SQL schema to ontologies, they all depend on the richness of the SQL schema with respect to its encoding of domain semantics. Any skepticism of the quality of the putative ontology is definitely well founded with respect to many legacy databases. Foreign key constraints were introduced as part of the SQL93 standard. Clearly, all direct mapping systems depend on explicit expression of such constraints to achieve accurate results. However, starting a number of years ago, databases began to be designed using sophisticated data modeling tools that capture considerable domain semantics such as foreign keys and check constraints. If the schema has been developed with such tools and in a normalized manner, the schema can portray enough semantics to create a semantically rich ontology. If this is the case, then direct mapping is the easiest way to create RDF from relational database data, by virtue of being completely automated.

On the other hand, the amount of domain semantics captured in SQL-DDL models is highly variable and occasionally does not contain rich structure. If these types of models are used in a direct mapping system, the putative ontology that is derived will lack rich semantics. Even though the putative ontology may not have the same expressivity as a domain ontology, it is not incorrect, hence the name putative. The deficiencies of a putative ontology become evident when an ontology produced by the system is compared with an ontology produced by a domain expert.

We assert that the scope and frequency of success of direct mapping transformation systems is very unlikely to achieve the scope that database–ontology mapping methods can achieve, however we consider direct mapping the first step in a two-step process. Sequeda *et al*. (2009) presents an application of direct mapping a relational database and manually mapping the putative ontology to a domain ontology, in order to overcome database heterogeneity. Direct mapping converts the relational database to ontology mapping problem to an ontology-to-ontology mapping problem. Therefore, we consider that the next step in this area is to bridge the gap between putative and domain ontologies.

In order to bridge this gap, we propose to refine the putative ontology through a bootstrapping architecture (Sequeda *et al*., 2008). This refinement process consists of initially matching the putative ontology with an existing domain ontology. Due to the fact that the putative ontology

[1] http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[2] http://www.w3.org/2001/sw/sweo/

[3] http://www.w3.org/2009/08/rdb2rdf-charter.html

lacks richer semantics and may have some ambiguity with respect to the domain ontology, we treat the alignment between the putative and domain ontology as an alignment hypothesis. Querying and analyzing the relational database data can test the alignment hypothesis. After the hypothesis has been tested, the new knowledge is used to refine the putative ontology. For example, a putative ontology may not represent inheritance, but its existence could be determined by the mapping to a domain ontology and the analysis of the content of the relational database. In this process, the putative ontology would be automatically matched to other domain ontologies. This would be an iterative process with different domain ontologies, until the putative ontology has been refined and it completely represents the domain semantics of the relational database.

Acknowledgments

This research is funded in part by the National Science Foundation Grant IIS-0531767, DBI-0851052, National Science Foundation Graduate Research Fellowship and the Spanish fundamental R&D Project myBigData.

References

Aho, A., Lam, M., Sethi, R. & Ullman, J. 2006. *Compilers: Principles, Techniques and Tools*. Addison-Wesley.

An, Y., Borgida, A. & Mylopoulos, J. 2005. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *Proceedings of On The Move to Meaninful Internet Systems (OTM'05): CoopIS, DOA, and ODBASE, Agia Napa, Cyprus*. Springer Verlag.

An, Y., Mylopoulos, J. & Borgida, A. 2006. Building semantic mappings from databases to ontologies. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI-06) Nectar Track*, Boston, MA, USA.

Arens, A., Chee, C. Y., Hsu, C. & Knoblock, C. 1993. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems* **2**(2), 127–158.

Arenas, M., Prud'hommeaux, E. & Sequeda, J. (ed.) 2010. *A direct mapping of relational data to RDF*. Retrieved February 17, 2011, from W3C Working Draft, 18 November 2010. `http://www.w3.org/TR/rdb-direct-mapping/`.

Astrova, I. 2004. Reverse engineering of relational databases to ontologies. In *Proceedings of 1st European Semantic Web Symposium (ESWS 2004)*, Crete, Greece. Springer.

Astrova, I., Korda, N. & Kalja, A. 2007. Rule-based transformation of SQL relational databases to OWL ontologies. In *Proceedings of 2nd International Conference on Metadata & Semantic Research*, Corfu Island, Greece.

Barrasa, J. & Gomez-Perez, A. 2006. Upgrading relational legacy data to the semantic web. In *Proceedings of 15th International World Wide Web Conference*. ACM, 1069–1070.

Barrasa, J., Corcho, O. & Gomez-Perez, A. 2004. R2O, an extensible and semantically based database-to-ontology mapping language. In *Proceedings of 2nd Workshop on Semantic Web and Databases*, Toronto, Canada.

Bizer, C. & Seaborne, A. 2004. D2RQ—treating non-RDF databases as virtual RDF graphs. In *Proceedings of 3rd International Semantic Web Conference*, Hiroshima, Japan.

Bizer, C., Heath, T. & Berners-Lee, T. 2009. Linked data – the story so far. *International Journal on Semantic Web and Information Systems* **5**(3), 1–22.

Brickley, D. & Guha, R. V. (ed.). 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. Retrieved February 17, 2011, from W3C Recommendation. `http://www.w3.org/TR/rdf-schema/`

Buccella, A., Penabad, M. R., Rodríguez, F. J., Fariña, A. & Cechich, A. 2004. From relational databases to OWL ontologies. In *Proceedings of the 6th Russian Conference on Digital Libraries*, Pushchino, Rusia.

Ceri, S. & Widom, J. 1993. Managing semantic heterogeneity with production rules and persistent queues. In *Proceedings of 19th Very Large Databases Conference*, Dublin, Ireland.

Chiang, R. H. L., Barron, T. M. & Storey, V. C. 1994. Extracting domain semantics for knowledge discovery in relational databases. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, Seattle, Washington, July 1994. Technical Report WS-94-03, AAAI Press, ISBN 0-929280-73-3, 473.

Chen, H., Wang, Y., Wang, H., Mao, Y., Tang, J. & Zhou, C. , *et al.* 2006. Towards a semantic web of relational databases: a practical semantic toolkit and an in-use case from traditional chinese medicine. In *Proceedings of the 5th International Semantic Web Conference*, Athens, GA, USA.

Collet, C., Huhns, M. N., Hsu, C. & Shen, W. 1991. Resource integration using a large knowledge base in Carnot. *Computer* **12**(24), 55–62.

Das, S., Sundara, S. & Cyganiak, R. (ed.) 2010. *R2RML: RDB to RDF Mapping Language*. Retrieved February 17, 2011, from W3C Working Draft, 28 October 2010. http://www.w3.org/TR/r2rml/

Dean, M. & Schreiber, G. (ed.) 2004. *OWL Web Ontological Language Reference*. Retrieved February 5, 2009, from W3C Recommendation. http://www.w3.org/TR/owl-ref/

de Laborda, C. P. & Conrad, S. 2005. Relational. OWL: a data and schema representation format based on OWL. In *Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling*, Newcastle, Australia.

Drummond, N. & Shearer, R. 2006. The Open World Assumption. *eSI Workshop: The Closed World of Databases meets the Open World of the Semantic Web*.

Du, H. & Wery, L. 1999. Micro: a normalization tool for relational database engineers. *Journal of Network and Computer Applications* **22**(4), 215–232.

Duo, D., Pan, J., Qin, H. & LePendu, P. 2006. Towards populating and querying the Semantic Web. In *Proceedings of the 2nd International workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, Athens, GA, USA.

Garcia-Molina, H., Ullman, J. & Widom, J. 2009. *Database Systems: The Complete Book*. Pearson Prentice Hall.

Hartig, O., Bizer, C. & Freytag, J. C. 2009. Executing SPARQL queries over the web of Linked Data. In *Proceedings of the 8th International Semantic Web Conference (ISWC2009)*, Chantilly, VA, USA.

Hayes, P. (ed.) 2004. *RDF Semantics*. Retrieved February 5, 2009, from W3C Recommendation. http://www.w3.org/TR/rdf-mt/

He, B., Patel, M., Zhang, Z. & Chang, K. 2007. Accessing the deep web. *Communications of the ACM* **50**(5), 94–101.

Klyne, G. & Carroll, J. J. (ed.) 2004. *Resource Description Framework (RDF): concepts and abstract syntact*. Retrieved February 17, 2011, from W3C Recommendation. http://www.w3.org/TR/rdf-concepts/

Korotkiy, M. & Top, J. 2004. From Relational Data to RDFS models. In *Proceedings of the 2004 International Conference on Web Engineering*, Munich, Germany.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M. & Smith, J. 2002. UML for ontology development. *Knowledge Engineering Review* **12**(1), 61–64.

Laclavik, M. 2006. RDB2Onto: Relational Database Data to Ontology Individual Mapping. In *Proceedings of Tools for Acquisition, Organisation and Presenting of Information and Knowledge*, Nízke Tatry, Slovaki.

Li, M., Du, X. & Wang, S. 2005. Learning ontology from relational database. In *Proceedings of the 4th International Conference on Machine Learning and Cybernetics*, Guangzhou, China.

Lubyte, L. & Tessaris, S. 2009. Automatic extraction of ontologies wrapping relational data sources. In *Proceedings of the 20th Database and Expert Systems Applications (DEXA2009)*, Linz, Austria.

Lutz, C., Areces, C., Horricks, I. & Satler, U. 2004. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research* **23**, 667–726.

Meier, D. 1983. *The Theory of Relational Databases*. Computer Science Press.

Miller, R., Haas, L. & Hernandez, M. 2000. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Database*, Cairo, Egypt.

Motik, B., Horrocks, I. & Sattler, U. 2007. Bridging the gap between OWL and relational databases. In *Proceedings of the 16th International Conference on World Wide Web*, Banff, Canada.

Noy, N. & Rector, A. (ed.) 2006. *Defining N-ary Relations on the Semantic Web*. W3C Working Group Note. http://www.w3.org/TR/swbp-n-aryRelations/

Pratt, P. J. 1990. *A Guide to SQL*. Boyd & Fraser Publishing Company.

Pratt, P. J. 1995. *A Guide to SQL*, 3rd edn. Boyd & Fraser Publishing Company.

Prud'hommeaux, E. & Seaborne, E. (eds) 2008. *SPARQL Query Language for RDF*. Retrieved February 17, 2011 from W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/

Sequeda, J. F., Tirmizi, S. H. & Miranker, D. P. 2007. SQL databases are a moving target. Position Paper for *W3C Workshop on RDF Access to Relational Databases*, Cambridge, MA, USA.

Sequeda, J. F., Tirmizi, S. H. & Miranker, D. P. 2008. A bootstrapping architecture for integration of relational databases to the Semantic Web. In *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference*, Karlsruhe, Germany.

Sequeda, J. F., Garcia-Castro, A., Corcho, O., Tirmizi, S. H. & Miranker, D. P. 2009. Overcoming database heterogeneity to facilitate social networks: the Colombian displaced population as a case study. In *Proceedings of the 18th International Conference on World Wide Web Ibero-America Track*, Madrid, Spain.

Shen, G., Huang, Z., Zhu, X. & Zhao, X. 2006. Research on the rules of mapping from relational model to OWL. In *Proceedings of the Workshop on OWL: Experiences and Directions*, Athens, GA, USA.

Smith, M. K., Welty, C. & McGuinness, D. L. (ed.) 2004. *OWL web ontology language guide*. Retrieved February 5, 2009, from W3C Recommendation. http://www.w3.org/TR/owl-guide/

SQL3 (ISO-ANSI Working Draft). (n.d.). Retrieved February 5, 2009, from http://www.inf.fu-berlin.de/lehre/SS94/einfdb/SQL3/sqlindex.html

Stonebraker, M. 1986. Triggers and inference in database systems. In *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Brodie, M. L. & Mylopoulos, J. (eds). Springer Topics in Information Systems. Springer-Verlag, 297–314.

Stojanovic, L., Stojanovic, N. & Volz, R. 2002a. Migrating data-intensive web sites into the Semantic Web. In *Proceedings of the 2002 ACM Symposium on Applied Computing, SAC'02*, Madrid, Spain, March 11–14, 2002, ACM, 1100–1107.

Stojanovic, L., Stojanovic, N. & Volz, R. 2002b. A reverse engineering approach for migrating data-intensive web sites to the Semantic Web. In *Proceedings of the IFIP 17th World Computer Congress—Tc12 Stream on Intelligent Information Processing*, August 25–30, 2002, Musen, M. A., Neumann, B., & Studer, R. (eds). IFIP Conference Proceedings, Kluwer B. V., Deventer, the Netherlands, **221**, 141–154.

Svihla, M. & Jelinek, I. 2004. Two Layer Mapping from Database to RDF. In *Proceedings of Electronic Computers and Informatics (ECI)*. Slovakia, Kosice.

Tirmizi, S. H., Sequeda, J. F. & Miranker, D. P. 2008. Translating SQL applications to the semantic web. In *Proceedings of the 19th International Conference on Database and Expert Systems Applications*, September 01–05, 2008, Turin, Italy, Bhowmick, S. S., Küng, J. & Wagner, R. (eds). Lecture Notes in Computer Science. Springer-Verlag, **5181**, 450–464.

Tirmizi, S. H., Aitken, S., Moreira, D., Mungall, C., Sequeda, J. F., Shah, N. H. & Miranker, D. P. 2011. Mapping between the OBO and OWL ontology languages. *Journal of Biomedical Semantics* **2**(Suppl 1), S3.

Vrandecic, D. & Sure, Y. 2008. How to design better ontology metrics. In *Proceedings of the 4th European Conference on Semantic Web: Research and Applications*, Innsbruck, Austria, June 03–07, 2007, Franconi, E., Kifer, M. & May, W. (eds), Lecture Notes In Computer Science. Springer-Verlag, **4519**, 311–325.

Wang, S., Shen, J. & Hong, T. 2000. Mining fuzzy functional dependencies from quantitative data. *IEEE International Conference on Systems, Man and Cybernetics* **5**, 3600–3606.

Biron, P. V., Permanente, K. & Malhotra, A. (ed.). 2004. *XML Schema Part 2: Datatypes*, 2nd edn. Retrieved February 5, 2009, from W3C Recommendation. `http://www.w3.org/TR/xmlschema-2/`

Xu, Z., Zhang, S. & Dong, Y. 2006. Mapping between Relational Database Schema and OWL Ontology for Deep Annotation. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web intelligence*. IEEE Computer Society, 548–552.