Relational.OWL - A Data and Schema Representation Format Based on OWL

Cristian Pérez de Laborda **Stefan Conrad**

Institute of Computer Science
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Germany,
Email: {perezdel, conrad}@cs.uni-duesseldorf.de

Abstract

One of the research fields which has recently gained much scientific interest within the database community are Peer-to-Peer databases, where peers have the autonomy to decide whether to join or to leave an information sharing environment at any time. Such volatile data nodes may appear shortly, collect or deliver some data, and disappear again. It even can not be assured that a peer joins the network ever again. In this paper we introduce a representation format fort both, schema and data information based on the Web Ontology Language OWL. According to the advantages of the Semantic Web we are thus able to represent and to transfer every schema and data component of a database to any partner, without having to define a data and schema exchange format explicitly.

Keywords: Data Representation, Schema Representation, Semantic Web, Web Ontology Language (OWL), Resource Description Framework (RDF), Relational Databases, Ontologies

1 Introduction

In this paper we introduce a *Web Ontology Language (OWL)*-based (Miller & Hendler 2004) representation format for relational data and schema components, which is particularly appropriate for exchanging items among remote database systems. OWL, originally created for the Semantic Web enables us to represent not only the data itself, but also its *interpretation*, i.e. knowledge about its format, its origin, its usage, or its original embedment in specific frameworks.

Hence, remote databases are instantly able to understand each other without having to arrange an explicit exchange format - the usage of OWL on both sides is sufficient. This would be impossible using present XML formats. The broad application field of this proposal includes all types of (multi)database systems (Litwin & Abdellatif 1986), e.g. Peer-to-Peer or Peer-to-Multi-Peer databases, where component databases share schema and data information (Halevy, Ives, Mork & Tatarinov 2003).

Contrary to present approaches where RDF is stored in relational databases, and relational data into RDF (Melnik 2001), this paper aims at bringing together the representation of both, database data and schema components with a common mediated language based on OWL, the powerful Semantic Web language recently recommended by the World Wide Web Consortium (McGuinness & van Harmelen 2004). In other words, we have created a representation format for data and schema items originally stored in databases, which is based on OWL's knowledge representation techniques (Broekstra, Klein, Decker, Fensel, van Harmelen & Horrocks 2001). In this paper we restrict to data originated from relational data sources, although we are basically able to represent items from any type of database. An application to these is subject to future work.

Adopting the opportunities given by OWL and our novel *Relational.OWL* ontology, we are now able to uniformly describe and share the schema of virtually any (relational) database, no matter from which vendor it was. Thereupon the schema representation itself can be used as an OWL ontology, to base the representation of the actual data on. As a result we obtain a model with three layers (Figure 1). On the topmost level, the most abstract one, is Relational.OWL. The layer underneath stands for an ontology, which was created using Relational.OWL in order to represent the schema of a specific relational database. The concrete data representation, based on the second ontology is placed on the lowest layer. Since we want to represent all three layers using OWL, we need the entire language (i.e. *OWL Full*) and not one of its sublanguages *OWL Lite* or *OWL DL.*

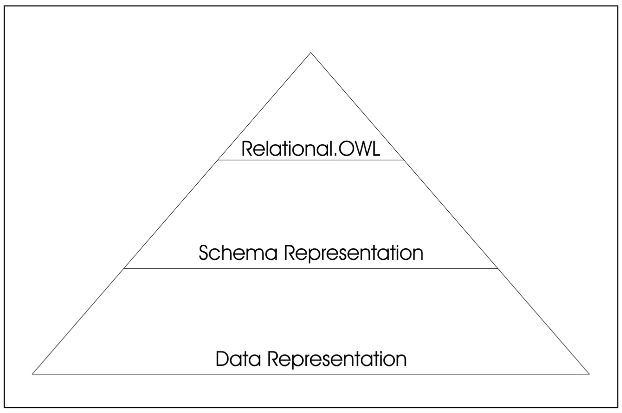

Figure 1: Three layers of abstraction using Relational.OWL

The remainder of this paper is organized as follows: After motivating our approach in section 2, we introduce Relational.OWL, our OWL ontology for representing the schema and data of a relational database in section 3. In section 4 we show how to represent the data of a database using its own specific OWL ontology. Section 5 catches up some related work, whereas section 6 concludes.

2 Motivation

In this section we present one of the main possible application fields of Relational.OWL, where a relational data and schema representation based on the techniques provided by OWL can improve the data sharing process substantially.

One of the research fields which has recently gained much scientific interest within the database community are Peer-to-Peer (P2P) databases and its derivatives like (Halevy et al. 2003), (Pérez de Laborda, Popfinger & Conrad 2004), or (Nejdl, Wolf, Qu, Decker, Sintek, Naeve, Nilsson, Palmér & Risch 2002). Having different aims to achieve, these projects are all based on the same principle: data stored on one single peer has to be made accessible to other remote peers and vice versa. Afterwards this data can be requested, queried, replicated, or integrated depending on the purpose of the remote system.

Sharing relational data in such environments is a challenging task, since:

Volatile Peers: Peers of a P2P network are usually autonomous. This autonomy includes the right to decide whether to join or to leave an information sharing environment at any time. Such volatile peers may appear shortly, collect or deliver some data, and disappear again. It even can not be assured that a peer joins the network ever again. Due to the short availability of potentially any peer, the negotiation of an exchange protocol for data and schema items becomes quite challenging. An exchange format, which can be understood instantly by all exchange partners would be more useful.

Data Distribution: Due to the characteristics of P2P environments, data which is significant for a peer may be spread over numerous data sources. Thus, this peer is required to collect that information from several remote data sources. In order to be able to receive and understand such data, the exchange partners need to arrange a data and a schema representation format. At worst, a peer would have to interpret a different format for every single data flow causing an unmanageable situation.

Relational Data: In classical filesharing networks, where textual or binary data (e.g. music) is offered, a file contains all the information required for understanding this data. Whereas sharing relational data means to offer data enriched with vital schema information including important instructions on how to interpret and use that data correctly (e.g. table and column names, consistency constraints, etc.). This meta data has to be transferred separately as long as data and meta data shall not be mashed.

Data Evolution: Data distributed over classical filesharing networks does usually not change, i.e. a file containing a music song will be identical, no matter how much time has passed. Whereas relational data evolves constantly resulting in changes concerning both, data and schema components. This fact has to be taken into account within the relational data sharing process.

As a result, sharing relational data within a Peer-to-Peer environment means to distribute not only data items themselves, but also their schemata among multiple previously unknown peers. We thus need an exchange format, which on the one hand can be understood by a broad community of peers without being explicitly arranged beforehand and which on the other hand has to be suitable for representing relational schemata and their corresponding data instances.

In this paper we present the *Relational.OWL* ontology, a promising representation method for relational schema and its corresponding data items, based on widely accepted knowledge representation techniques. It is a favorable fundament for a broad application field ranging from data exchanges over backup mechanisms to relational database management systems, which store their data internally using this technique.

Relational.OWL is not a classical exchange format, but a representation technique, which is equally suitable for data backups or migration scenarios. Anyway, *Relational.OWL* itself may easily be extended to fit all requirements of a data exchange format including modification, addition, or deletion directives.

In fact, including this information into our data and schema representation would mean to loose the independence from the application fields, i.e. such an exchange format could not be used for a backup any more. We thus have decided not to include exchange or replication specific instructions into *Relational.OWL*, but to keep it an application independent representation technique.

3 Relational.OWL

In this section we describe the advantages of using OWL as representation language for databases within an information sharing environment and analyze, which parts of relational metadata have to be included into our Relational.OWL ontology. A short representation of an actual database schema concludes this section.

3.1 Reasons for OWL

The representation of relational data and schema with the Web Ontology Language OWL entails several advantages over classical (semi)structured exchange formats like XML (Bray, Paoli & Sperberg-McQueen 1997). In this section we discuss these advantages and explain, why the usage of OWL should be considered, although a minor increase of data overhead has to be taken into account.

Knowledge Representation: The knowledge representation approach of OWL enables us to write formal conceptualizations of domain models, the so-called ontologies (Antoniou & van Harmelen 2003). Having created such an ontology we are able to encode knowledge about things and their interrelationships within our specific domain into a machine-understandable format, which can afterwards be decoded and interpreted by any remote peer who has access to that ontology.

Applied to the domain of relational databases, we can describe data and schema items and its corresponding interconnections in a machine-processable and understandable way, as soon as we have defined an ontology for the representation of relational data(bases).

Reliable Data and Schema Exchange: The only way to guarantee the faultless interpretation of data and schema items on a remote node, is knowledge representation. The knowledge representation process prevents different sites (e.g. data exchange partners) to interpret the same data differently. Thus, it can be guaranteed that an item exchanged among remote peers will always maintain precisely its intended meaning.

The risk of a possible misunderstanding can usually be minimized through a face to face communication or previous agreement between the exchange partners. Nevertheless, this can not be accomplished within a volatile environment, where peers may appear and disappear at any time (e.g. P2P databases). In this case it is vital to have a representation format which is unambiguous for all exchange partners involved.

No Explicit Exchange Format: It is very sophisticated to arrange a common representation format for a data exchange, especially if the partners involved barely know each other and the schema constantly changes. In the latter situation, a communication channel set up by two nodes may probably be used only once, thus the arrangement of a proprietary format would be a tremendous overhead.

Although it is possible to arrange such formats (in)formally, this leads to unmanageable amounts of representation formats, particularly if a node is involved in several data exchanges. As we have discussed above, the usage of a (semi)structured format in its classical way could cause misunderstandings among the sites involved. Thus, using knowledge representation techniques enables remote peers to understand the information provided without having to arrange a specific exchange or representation format, since it is provided with OWL (e.g. the OWL XML representation (Hori, Euzenat & Patel-Schneider 2002)). The only requirements for establishing such a substantial communication are components capable to handle OWL and a common ontology like *Relational.OWL*, which is presented in this paper.

Convertible Representation: One of the main advantages of using common knowledge representation techniques is the simple interconnectivity of existing ontologies. Two communities using different ontologies for the representation of relational databases could easily collaborate, as soon as a semantic mapping between these ontologies is created (Doan, Madhavan, Domingos & Halevy 2002).

Having such a mediator ontology, both communities are instantly able to understand each others' representation format, without having to change a single thing on their data and schema import or export processes. The interpretation of mediator ontologies is an integral part of the knowledge representation techniques used in OWL.

Data as an Instance of its Schema: Given the fact that OWL enables the creation of classes and its instances with one and the same syntax, we are able to describe relational schema and data items with OWL. Furthermore we link schema and data representation in a singular way resulting in a homogenous data and schema format, where data items are defined as instances of their own schemata. This representation corresponds exactly to the internal representation used by current relational database systems.

Uniform Representation: Since we have to describe data and on a higher abstraction level its schema, we require two different representation formats, especially if we want to have the data represented as an instance of its schema. OWL supports both, ontological modeling and reasoning using the same syntax. We thus can provide an uniform framework for the precise representation of relational schema and its data items. Contrary to this, other languages like RDF or XML need to fall back to their corresponding modeling languages called RDF Schema, XML Schema, or Document Type Definition (DTD).

Reasonable Data Overhead: Due to the characteristics mentioned above, especially concerning the powerful ontological meaning and reasoning implemented in OWL, we have designed a promising technique for the representation of relational data and schemata, which is more powerful than the established representation or exchange formats like genuine XML or RDF. In order to achieve this extended functionality we have to take an increased amount of data into account.

First implementations have shown that data overhead is increased by about 20% compared to the `Rec2XML` scalar function in IBM's UDB, which generates XML data from a relational query (Figure 2) . This fact seems to be a major drawback, but is negligible, since data resulting from Relational.OWL can be compressed with a higher rate using common compression algorithms (e.g. *ZIP*). Further evaluations have revealed that the size of both files differ only in about one percent after compression, since the representation with Relational.OWL contains recurring parts and thus can be compressed using a higher compression rate.

```
<...>
<country>
  <column name="COUNTRYID">32</column>
  <column name="NAME">Deutschland</column>
</country>
<country>
  <column name="COUNTRYID">152</column>
  <column name="NAME">España</column>
</country>
</ ...>
```

Figure 2: XML data generated with `Rec2XML`

Concluding, we have decided to use OWL for the representation of relational data and schema items, since it provides a feasible balance between a powerful knowledge representation technique and a reasonable amount of data overhead. Applying knowledge representation techniques to the field of data and schema extraction results in various advantages, which could have a big impact especially within the P2P databases, where volatile peers may appear for a short time offering or demanding for data. As long as all partners understand the Web Ontology Language OWL, we do not need to negotiate a data or schema exchange format any more, since all partners involved are capable to talk the same *language* Relational.OWL.

3.2 Relevant Metadata

A database management system maintains a huge amount of metadata information to manage the whole system in a proper way. In current relational database systems, this information is stored in predefined system-tables, also called *Data Dictionary* or *Repository*.

We have decided to include only the upmost relevant metadata into our Relational.OWL ontology, since a large amount of the data stored in the repository is very system-specific and thus not suitable for a semantic representation. The metadata we have selected is the indispensable one for a proper interpretation of the actual data (not metadata, cp. section 4) representation. In fact, the set of schema items described may easily be extended, if it is required later on. Hence we have included the following metadata items into the Relational.OWL ontology:

Tables and Columns As implied by its name, the most important schema component of a relational database is the relation, also called table. Additionally each table consists of columns (attributes), where the actual data is being stored. Both schema components are the upmost essential information for representing the schema of a relational database and thus have to be included into the Relational.OWL representation.

Primary and Foreign Keys One or more columns may compose the primary key of a table. This information can be very useful for a target system, in particular if data updates have to be synchronized. Otherwise problems could arise assigning the new values to the old ones. Analogous, Foreign Keys have to be represented, otherwise the data on the target system could become inconsistent.

Data Types Data types restrict the possible values in a column (e.g. only `integer` or only `varchar` values). This special form of consistency constraints can be indispensable if a bidirectional data synchronization is being performed and very useful for performing a small consistency check of the data received by the target system.

OWL provides built-in datatypes and the possibility to fall back to the XML Schema datatypes (Biron & Malhotra 2001). Since there is no standard way to use the latter within OWL (Patel-Schneider & Horrocks 2004), we have decided to restrict this first version of Relational.OWL to those datatypes clearly defined within the OWL abstract syntax. Nevertheless, we need a technique to represent possible restrictions concerning the maximal length of values stored in each column (e.g. `varchar(100)`).

Concluding, it is necessary to include a representation for

- tables,
- columns,
- datatypes possibly with length restrictions,
- primary keys,
- foreign keys, and
- the relations among each other

into the Relational.OWL ontology.

3.3 The Relational.OWL Ontology

As concluded in paragraph 3.2, we require an OWL ontology which describes the schema of a relational database in an abstract way. This OWL representation can easily be interpreted by any remote database or application, which is capable to process OWL and has access to the Relational.OWL ontology. As a further step we use this schema representation itself as a novel ontology for creating a representation format, which is suitable for the corresponding data items.

To describe the schema of a relational database with the techniques provided by the Web Ontology Language OWL, we have to define reference OWL classes centrally, to which any document describing such a database can refer to. The abstract representation of classes like `Table` or `Column` become hereby a central part of the knowledge representation process realized within OWL. Additionally we have to specify possible relationships among these classes resulting in an ontology, a relational database can easily be described with. We call this central representation of abstract schema components and relationships *Relational.OWL*.

Similar representations based on RDF or OWL, which may evolve elsewhere, may be linked to Reltional.OWL with corresponding `owl:equivalentClass` or `owl:equivalentProperty` relationships. As a result, database representations using one of the ontologies mapped, can be understood by any application, which usually uses one of these ontologies.

In other words, each component (database) involved in a representation based on one of these ontologies is able to process documents based on any of the interconnected representation formats. We do not even have to adapt the reasoning processes, since it is enough to create a semantic mapping between two or more ontologies to make them exchangeable, as long as they correlate semantically.

In the following we describe the components of the Relational.OWL ontology, our proposal for semantic reasoning in data and schema exchanges. We can create system specific schema representations based on this ontology, which themselves can be used as ontologies for the representation of data items. In the remainder of this paper we abbreviate our main namespace http://www.dbs.cs.uni-duesseldorf.de/RDF/relational.owl# with the prefix `dbs`. `Rdf`, `rdfs`, and `owl` correspond to the commonly used prefixes for RDF, RDF Schema, and OWL.

A summary of all the classes represented in the Relational.OWL ontology is provided in Table 1. Table 2 contains a list of the relationships, which interconnect these classes. The exact class and property definitions can be accessed online at the URI specified above.

As mentioned above, we did not include a representation of all possible meta information into our ontology. Hence, items like indexes, triggers, or tablespaces are not considered, but may easily be included in a future version of Relational.OWL. In fact, the part of the relational schema we have chosen to describe is sufficient to represent the whole actual data stored in that database. Additional extensions in the Relational.OWL ontology would only increase the schema data overhead.

3.4 Example

This section provides an example on how to represent the schema of existing databases using Relational.OWL as their original ontology. The snippet in Figure 3 is derived from the schema representation of a collegial database, which contains personal information and further collegiate data of students.

The first element corresponds to a table, which contains the residence of a student. In this case, the `ADDRESS` ID is equivalent to the name of the table in the original database. Instead of exclusively using the table name as an identifier, a complete URI pointing at this specific table can be spec-

rdf:ID	rdfs:subClassOf	rdfs:comment
dbs:Database	rdf:Bag	The class of databases.
dbs:Table	rdf:Seq	The class of database tables.
dbs:Column	rdfs:Resource	The class of database columns.
dbs:PrimaryKey	rdf:Bag	The Primary Key of a Table.

Table 1: Classes defined in the Relational.OWL ontology

rdf:ID	rdfs:domain	rdfs:range	rdfs:comment
dbs:has	owl:Thing	owl:Thing	A Thing can have other Things inside.
dbs:hasTable	dbs:Database	dbs:Table	A Database has a set of Tables.
dbs:hasColumn	dbs:Table	dbs:Column	A Table has a set of Columns.
dbs:isIdentifiedBy	dbs:Table	dbs:PrimaryKey	A Table is identified by a Primary Key.
dbs:references	dbs:Column	dbs:Column	Foreign Key rel.ship between Columns.
dbs:length	dbs:Column	xsd:nonNegativeInteger	Maximal length of an entry in that Column.
dbs:scale	dbs:Column	xsd:nonNegativeInteger	The scale an entry of the Column may have.

Table 2: Properties defined in the Relational.OWL ontology

```
<...>
 <owl:Class rdf:ID="ADDRESS">
  <rdf:type rdf:resource="&dbs;Table"/>
  <dbs:hasColumn rdf:resource="#ADDRESS.ADDRESSID"/>
  <dbs:hasColumn rdf:resource="#ADDRESS.STREET"/>
  <dbs:hasColumn rdf:resource="#ADDRESS.ZIP"/>
  <dbs:hasColumn rdf:resource="#ADDRESS.CITY"/>
  <dbs:hasColumn rdf:resource="#ADDRESS.COUNTRYID"/>
  <dbs:isIdentifiedBy>
   <dbs:PrimaryKey>
    <dbs:hasColumn rdf:resource="#ADDRESS.ADDRESSID"/>
   </dbs:PrimaryKey>
  </dbs:isIdentifiedBy>
 </owl:Class>

 <owl:DatatypeProperty rdf:ID="ADDRESS.ZIP">
  <rdf:type rdf:resource="&dbs;Column"/>
  <rdfs:domain rdf:resource="#ADDRESS"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <dbs:length>8</dbs:length>
 </owl:DatatypeProperty>

 <owl:DatatypeProperty rdf:ID="ADDRESS.COUNTRYID">
  <rdf:type rdf:resource="&dbs;Column"/>
  <rdfs:domain rdf:resource="#ADDRESS"/>
  <dbs:references rdf:resource="#COUNTRY.COUNTRYID"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
 </owl:DatatypeProperty>
</ ...>
```

Figure 3: Schema represented with OWL and Relational.OWL

ified using identifiers like in (Pérez de Laborda & Conrad 2003). The five columns of this table are defined with `owl:DatatypeProperty` classes, where all the properties required are specified. Both objects are linked using a `dbs:hasColumn` property.

The primary key property of the table is represented with the `dbs:isIdentifiedBy` property, whereas the `dbs:PrimaryKey` Object corresponds the actual primary key. Since the primary key itself may consist of more than one column, the corresponding list can be specified with `dbs:hasColumn` entries. The second element describes the ZIP column in the address table, e.g. a column may contain string values with a maximum length of eight characters. The foreign key (`dbs:references`) from the `ADDRESS.COUNTRYID` column to the ID in the country table can be found in the second `owl:DatatypeProperty` element.

The complete example showing all features of the knowledge representation using Relational.OWL as an ontology can be found at http://www.dbs.cs.uni-duesseldorf.de/RDF/schema.owl.

4 Data Representation

After the description of how to represent the schema of a database using OWL and our Relational.OWL ontology, we now focus on how to use the schema representation just created as a novel ontology. With this tailored ontology-based representation of the database schema, we are able to represent the data, which is stored in that specific database. As a result, data stored in a relational database can be represented as instances of its own OWL schema.

4.1 A novel Ontology

According to the possibilities given by OWL and due to the schema representation presented above, we are able to use individuals (i.e. instances) of our Relational.OWL ontology as classes. Thus the schema representation just created belongs to OWL Full, and not to OWL Lite, nor to OWL DL (Dean & Schreiber 2004). Of course, the fact that we can not restrict the complexity to one of the subclasses does not automatically result in a complex representation of data and schema items. First implementations make us confident of most OWL reasoning tools being able to handle data and schema representations created using Relational.OWL.

In order to realize this kind of data representation process, we have to ensure that all components involved (e.g. exchange partners)

- are able to process and understand OWL,
- have access to Relational.OWL or a semantically equivalent ontology, and
- have access to the OWL schema representation of the corresponding database.

Using the schema constructed above as a novel ontology means to represent the data stored in that specific database using a tailored data representation technique. As a result, the data can be handled using common OWL techniques for data backups, data exchanges, or any kind of data processing tasks.

In fact, this interdigitation of schema and data corresponds exactly to the data management in relational database systems, where data items are stored as instances of their schema. As a result, the ontology, the data is described with, changes as soon as the schema of the originating database is altered. Using conventional techniques in data exchange processes would mean to manually adjust the corresponding exchange format. Using knowledge representation techniques, this is done automatically.

4.2 Example

Besides the example presented in section 3.4, where we explained how to represent the schema of a database containing information concerning students, we are now able to describe the actual data stored in that specific database (Figure 4). The namespace `dbinst` points to the location where the OWL representation of the schema is stored. Hence, it is required either to hold a copy of the relevant schema file, or to have access to such a representation (local copy vs. local accessible copy).

The example contains four elements, where the first two represent entries in the address table and the latter two correspond to an entry in the country table respectively. The address dataset contains all the information described in the previous example, i.e. an ID, a Street, a ZIP code, a city, and a country ID. Since we are using OWL in its XML representation, we benefit from its sophisticated features concerning internationalization: Declaring the proper encoding ensures special characters (e.g. ä, ø, or ñ) to be interpreted correctly.

Having stored all required information concerning the structure of the database in the schema file shown in section 3.4, we do not need to indicate that the `ADDRESSID` corresponds to the primary key of the table any more (`dbinst:ADDRESS`). The same occurs with the primary key of the `COUNTRY` table. This information is available in the schema representation, which is accessible. It would be redundant to include it into the data representation.

The complete example containing a complete version of both, the schema and data files can be downloaded at http://www.dbs.cs.uni-duesseldorf.de/RDF/, just as our first prototype which allows exporting data and schema items stored in a relational database using OWL as representation language. The prototype was developed in Java using a JDBC-Bridge and tested with an IBM DB2 UDB v. 8.1 database. A new version based completely on Jena (*Jena - A Semantic Web Framework for Java* 2004) is going to be accessible soon.

5 Related Work

Relational.OWL was designed within the DÍGAME project (Pérez de Laborda et al. 2004), where we have created an architecture to support flexible intra- and inter-enterprise collaboration. This architecture enables the propagation of data and schema updates done actively over import/export-components between dynamically connectable data nodes.

Furthermore we have developed the Link Pattern Catalog (Popfinger, Pérez de Laborda & Conrad 2004) as a modeling guideline for recurring problems appearing during the design or description of information grids and P2P networks. Tightly coupled with Relational.OWL is also (Pérez de Laborda & Conrad 2003), where we introduced an identifier for items stored in relational databases, which is based on an early forerunner of the ontology we have presented in this paper.

Since the raise of XML in the late 1990s and early 2000s a large number of different exchange formats for relational database systems have been developed based on XML (e.g. Torque as part of the Apache DB Project (The Apache DB Project 2004)). Actually, each vendor of (object)relational database systems tried to establish its own XML representation, like ORACLE's `XMLElement` function or IBM's `Rec2XML` scalar function we already mentioned above. Since these different dialects can easily be converted using XSLT, this babylonic chaos of different representation languages is broadly accepted. Nevertheless, the transformation rules between the different dialects have to be created manually.

After Berners-Lee et al. had expressed their vision of the next generation Web, a Semantic Web in (Berners-Lee, Hendler & Lassila 2001), the community started to seriously adopt the idea of semantic reasoning within the World Wide Web. This includes also some ideas on how to extract data from (relational) database systems, whereof (Berners-Lee 1998) can be seen as an early forerunner. Nevertheless, this mapping of relational data to RDF is rather rudimental, lacking, e.g. of a concrete schema representation.

Bizer introduced in (Bizer 2003) a mapping language between relational data and RDF, particularly between specific relational query results and RDF. Contrary to our approach, *D2R MAP* converts the stored data into "real" RDF objects, i.e. an address would be represented as a RDF address object. This approach takes into account, that the original database cannot be reconstructed using this kind of data representation anymore, since it does not contain information concerning the original schema of the database. As a result, the data represented with the D2R MAP language looses its relationship to the original database. Tracing the data to its original storage position is thus hardly possible.

Semantic integration of corporate information resources is the main topic in (Barrett, Jones, Yuan, Sawaya, Uschold, Adams & Folger 2002), where Barrett et al. use RDF as a standardized communication language between all components. The main difference to our approach is also their mapping of data resulting from queries to existing ontologies, which describe real-world relationships among objects. Hence, their aim is not to represent the relationships as they are in the database, but as they are in real-world. The original database can not be rebuilt using this data, since this approach does not include a relational schema representation and the resulting data is not connected to the original data source any more.

```
<... />
  <dbinst:ADDRESS>
    <dbinst:ADDRESS.ADDRESSID>3248</dbinst:ADDRESS.ADDRESSID>
    <dbinst:ADDRESS.STREET>Universitätsstr. 1</dbinst:ADDRESS.STREET>
    <dbinst:ADDRESS.ZIP>40225</dbinst:ADDRESS.ZIP>
    <dbinst:ADDRESS.CITY>Düsseldorf</dbinst:ADDRESS.CITY>
    <dbinst:ADDRESS.COUNTRYID>32</dbinst:ADDRESS.COUNTRYID>
  </dbinst:ADDRESS>
  <dbinst:ADDRESS>
    <dbinst:ADDRESS.ADDRESSID>6824</dbinst:ADDRESS.ADDRESSID>
    <dbinst:ADDRESS.STREET>Paseo Manuel de Lardizabal, 1</dbinst:ADDRESS.STREET>
    <dbinst:ADDRESS.ZIP>20018</dbinst:ADDRESS.ZIP>
    <dbinst:ADDRESS.CITY>Donostia-San Sebastián</dbinst:ADDRESS.CITY>
    <dbinst:ADDRESS.COUNTRYID>152</dbinst:ADDRESS.COUNTRYID>
  </dbinst:ADDRESS>
  <... />
  <dbinst:COUNTRY>
    <dbinst:COUNTRY.COUNTRYID>32</dbinst:COUNTRY.COUNTRYID>
    <dbinst:COUNTRY.NAME>Deutschland</dbinst:COUNTRY.NAME>
  </dbinst:COUNTRY>
  <dbinst:COUNTRY>
    <dbinst:COUNTRY.COUNTRYID>152</dbinst:COUNTRY.COUNTRYID>
    <dbinst:COUNTRY.NAME>España</dbinst:COUNTRY.NAME>
  </dbinst:COUNTRY>
<... />
```

Figure 4: Example of Data represented with OWL based on its own Schema

6 Summary and Further Research

In this paper we have shown how to represent schema and data items originally stored in relational database systems using our own OWL ontology. Relational.OWL enables us to semantically represent the schema of any relational database. This representation itself can be interpreted, due to the properties of OWL Full, as a novel ontology. Based on the latter ontology, we can now semantically represent the data stored in this specific database.

The advantage of this representation technique is obvious: Both, schema and data changes can automatically be transferred to and processed by any remote database system, which is able to understand knowledge representation techniques used within OWL. Misunderstandings are impossible.

Besides the refinement and completion of the concrete schema representation, we consider on how to adopt our technique to other types of database systems. Similar solutions can easily be found for Object-Oriented Databases, Hierarchical Databases like IMS, or its hybrid the modern and more common X.500 or LDAP Directory Systems (Howes, Smith & Good 1999).

A further extension for Relational.OWL could be a corresponding protocol extending the possibilities of Relational.OWL to particularly support data exchanges or replications. There we could employ the advantages of our knowledge representation technique for recurring problems occurring within such a data exchange process, e.g. identifying the same data items on remote databases.

Although autonomously communicating databases in a metadata exchange are still more vision than reality, our model takes us one step further.

References

Antoniou, G. & van Harmelen, F. (2003), Web Ontology Language: OWL, *in* S. Staab & R. Studer, eds, 'Handbook on Ontologies in Information Systems', Springer-Verlag.

Barrett, T., Jones, D., Yuan, J., Sawaya, J., Uschold, M., Adams, T. & Folger, D. (2002), RDF Representation of Metadata for Semantic Integration of Corporate Information Resources, *in* 'International Workshop Real World and Semantic Web Applications 2002'.

Berners-Lee, T. (1998), 'Relational Databases and the Semantic Web (in Design Issues)', http://www.w3.org/DesignIssues/RDB-RDF.html.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001), 'The semantic web', *Scientific American* .

Biron, P. V. & Malhotra, A. (2001), 'XML schema part 2: Datatypes', http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

Bizer, C. (2003), D2R MAP-A Database to RDF Mapping Language, *in* 'WWW2003, The Twelfth International World Wide Web Conference', Budapest, HUNGARY. poster presentation.

Bray, T., Paoli, J. & Sperberg-McQueen, M. (1997), 'Extensible Markup Language (XML)', http://www.w3.org/TR/1998/REC-xml-19980210.

Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F. & Horrocks, I. (2001), Enabling Knowledge Representation on the Web by Extending RDF Schema, *in* 'Proceedings of the tenth international conference on World Wide Web', ACM Press, pp. 467–478.

Dean, M. & Schreiber, G. (2004), 'OWL Web Ontology Language Reference', http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Doan, A., Madhavan, J., Domingos, P. & Halevy, A. (2002), Learning to Map between Ontologies on the Semantic Web, *in* 'Proceedings of the eleventh international conference on World Wide Web', ACM Press, pp. 662–673.

Halevy, A. Y., Ives, Z. G., Mork, P. & Tatarinov, I. (2003), Piazza: Data Management Infrastructure for Semantic Web Applications, *in* 'Proceedings of the twelfth international conference on World Wide Web', Budapest, Hungary, pp. 556–567.

Hori, M., Euzenat, J. & Patel-Schneider, P. F. (2002), 'OWL Web Ontology Language XML Presentation Syntax', http://www.w3.org/TR/owl-xmlsyntax/.

Howes, T. A., Smith, M. C. & Good, G. S. (1999), *Understanding and Deploying LDAP Directory Services*, New Riders, Indianapolis.

Jena - A Semantic Web Framework for Java (2004), http://jena.sourceforge.net/.

Litwin, W. & Abdellatif, A. (1986), 'Multidatabase Interoperability', *Computer* **19**(12), 10–18.

McGuinness, D. L. & van Harmelen, F. (2004), 'OWL Web Ontology Language Overview', http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Melnik, S. (2001), 'Storing RDF in a relational database', http://www-db.stanford.edu/∼melnik/rdf/db.html.

Miller, E. & Hendler, J. (2004), 'Web Ontology Language (OWL)', http://www.w3.org/2004/OWL.

Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M. & Risch, T. (2002), EDUTELLA: A P2P Networking Infrastructure Based on RDF, *in* 'Proceedings of the Eleventh International Conference on World Wide Web', ACM Press, pp. 604–615.

Patel-Schneider, P. F. & Horrocks, I. (2004), 'OWL Web Ontology Language Semantics and Abstract Syntax Section 2. Abstract Syntax', http://www.w3.org/TR/2004/REC-owl-semantics-20040210/syntax.html.

Pérez de Laborda, C. & Conrad, S. (2003), A Semantic Web based Identification Mechanism for Databases, *in* 'Proceedings of the 10th International Workshop on Knowledge Representation meets Databases (KRDB 2003), Hamburg, Germany, September 15-16, 2003', Vol. 79 of *CEUR Workshop Proceedings*, Technical University of Aachen (RWTH), pp. 123–130.

Pérez de Laborda, C., Popfinger, C. & Conrad, S. (2004), DÍGAME: A Vision of an Active Multidatabase with Push-based Schema and Data Propagation, *in* 'Proceedings of the GI-/GMDS-Workshop on Enterprise Application Integration (EAI'04)', Vol. 93 of *CEUR Workshop Proceedings*.

Popfinger, C., Pérez de Laborda, C. & Conrad, S. (2004), Link Patterns for Modeling Information Grids and P2P Networks, *in* Il-Yeol Song, Stephen W. Liddle, Tok Wang Ling, and Peter Scheuermann, eds., 'Conceptual Modeling - ER 2004, 23rd International Conference on Conceptual Modeling, Shanghai, China, November 8-12, 2004 Proceedings', Vol. 3288 of *Lecture Notes in Computer Science*, Springer Verlag.

The Apache DB Project (2004), 'Torque', http://db.apache.org/torque/.