Database to Semantic Web Mapping using RDF Query Languages

Cristian Pérez de Laborda and Stefan Conrad

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

Abstract. One of the main drawbacks of the Semantic Web is the lack of semantically rich data, since most of the information is still stored in relational databases. In this paper, we present an approach to map legacy data stored in relational databases into the Semantic Web using virtually any modern RDF query language, as long as it is closed within RDF. Consequently, a Semantic Web developer does not need to learn and adopt a new mapping language, but he may perform the mapping task using his preferred RDF query language.

1 Motivation

Despite the vision of a Semantic Web [6] and many efforts helping to realize it, the actual Semantic Web still lacks of enough semantic data. Most information is still modeled and stored in relational databases and thus out of reach for many Semantic Web applications. As a consequence, such applications need to create a corresponding mapping between the relational and the semantic models by themselves for being able to access relational data. Realizing this situation, some efforts have arisen to straighten out this deplorable situation.
Most approaches translate relational data into a Semantic Web representation using a proprietary mapping language (cf. Section 2). In [19] we have introduced Relational.OWL, our technique to automatically transform relational data into a machine processable and understandable representation (cf. Section 3.1). Nevertheless, such a representation does not include *real* semantics, since it converts the schema of a database automatically into an ontology and the data items as its instances, i.e. the data is described as it was in the database. For many Semantic Web applications, this is a reasonable technique, since they are able to quickly access legacy data stored in a relational database using their own built-in functionality. However, such a representation could be inappropriate, if the data has to be processed for further reasoning tasks.
In this paper we present how to map relational data into the Semantic Web using virtually any modern RDF query language, as long as the language is closed within RDF, i.e. it returns valid RDF graphs as query results. For this purpose, data and schema components of the original relational database are

Conceptual Modeling - ER 2006, 25th International Conference on Conceptual Modeling, Tucson, Arizona, November 2006, Springer Verlag

first translated automatically into their Semantic Web representation based on Relational.OWL. Thereupon, they may either be processed or mapped directly to a target ontology. To perform such a mapping task, a Semantic Web developer does not need to learn and adopt a new mapping language, but he may perform the mapping task using his preferred RDF query language.
The remainder of this paper is organized as follows. In the next Section we discuss some related research. In Section 3 the foundations of this paper are described. The relational database to Semantic Web mapping process is introduced in Section 4 and evaluated in Section 5. Finally, we conclude in Section 6 with a short discussion and some ideas for future work.

2 Related Work

Recently, some efforts arose in bringing together relational databases and the Semantic Web. Nevertheless, most of these approaches do not use relational databases as a data source, but to store RDF triples in tailored tables, exploting the improved query performance of current relational databases (e.g. [13], [16], or [11]). The main drawback of such approaches is, that the corresponding data has to be available in RDF, i.e. their aim is not to convert legacy data into a Semantic Web representation, but to give applications fast access to RDF triples.
Some approaches try to map legacy relational databases to the Semantic Web. Bizer [7] for example, introduces a mapping from relational databases to RDF. Unlike our approach which is based on existing query languages, this method requires a specific mapping language, which, although it is based on RDF, still has to be learned and adopted by the corresponding developers.
An et al. outline in [5] a further approach from tables to ontologies. Unlike our technique, this approach maps database schemas directly into ontological concepts, assuming that the required database was designed following several ER design principles, e.g. the database is normalized and contains meaningful table or column names.
Petrini and Risch introduce in [21] their technique to query relational databases using RDF query languages, which is closely related to the approach presented in this paper. Nevertheless, it has some drawbacks. The mapping from relational tables to the Semantic Web is defined within a custom made mapping table, where columns or tables are related to objects or attribute values. As a result, the mappings between both worlds are always 1:1. Our mapping technique is completely based on the Semantic Web and allows the mappings to be as complex as a query language can be, i.e. we would even be able to use aggregations, if they are supported by the query language used.

3 Foundations

In this section we present the foundations, this work is based on. First, we introduce Relational.OWL, an approach to automatically transform relational data and schema items into a Semantic Web representation. After this, we explain in

Section 3.2 the difficulties in querying the resulting RDF graphs using current RDF query languages.

3.1 Relational.OWL

In [19] we introduced Relational.OWL, a data and schema representation, which adopts Semantic Web techniques to the data and schema representation process of (relational) databases. Contrary to other approaches where RDF is stored in relational databases (e.g. [15]), Relational.OWL aims at bringing together the representation of both, database data and schema components with a common mediated language, based on the *Resource Description Framework* (RDF) and the *Web Ontology Language* (OWL) [14]. In this section we give a short introduction to the Relational.OWL representation technique, since it is essential for a subsequent application to the mapping process presented in this paper.

The Relational.OWL Ontology To describe the schema of a relational database with the techniques provided by RDF and OWL, we have to define reference OWL classes centrally, to which any document describing such a database can refer. The abstract representation of classes like `Table` or `Column` becomes 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 the Relational.OWL ontology. It contains abstract definitions of relational databases $\mathfrak{D}$, tables $\mathfrak{T}$, columns $\mathfrak{C}$, primary keys $\mathfrak{P}$, foreign keys $\mathfrak{F}$, and their corresponding relationships.
For each relational database $\mathcal{RDB}_i$, a Semantic Web correspondent $\mathcal{ROWL}_i(\mathcal{S}_i, \mathcal{I}_i)$ is created, where $\mathcal{S}_i$ is the schema and $\mathcal{I}_i$ the data instance representation. $\mathcal{S}_i$ will usually contain one subclass $\mathcal{D}_i$ of $\mathfrak{D}$. Analogously, for each relation $\mathcal{R}_1, ..., \mathcal{R}_m \in \mathcal{RDB}_i$, a subclass $\mathcal{T}_1, ..., \mathcal{T}_m$ of $\mathfrak{T}$ is created and included into $\mathcal{S}_i$. The $\in$ relationship between $\mathcal{RDB}_i$ and $\mathcal{R}_j$ is then added using a corresponding *hasTable* property within the $\mathcal{D}_i$ class. The remaining components and their relationships are transformed correspondingly.
A snippet of a database representation using Relational.OWL is provided in Fig. 1. Its first element corresponds to a table containing residence information of a business contact. In this case, the `rdf:ID ADDRESS` is equivalent to the table name in the original database. Instead of exclusively using the table name as an identifier, a complete URI pointing at this specific table can be specified using an identifier, e.g. as in [17]. Each of the five columns is defined using a `owl:DatatypeProperty` class, where all the properties required are specified. The corresponding `&dbs;Table` and `&dbs;Column` objects are then linked using a `dbs:hasColumn` property.
The primary key property of the table is represented using a `dbs:isIdentifiedBy` property, whereas the `dbs:PrimaryKey` Object corresponds to the actual primary key. Since the primary key itself may consist of more than one column, they are specified with `dbs:hasColumn` entries. The second element in Fig. 1 describes the ZIP column of the address table. It contains string values with a maximum length of eight characters.

```
<...>
 <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>
</ ...>
```

Fig. 1. Schema Representation

Data Representation After having created a schema representation of a database $\mathcal{RDB}_i$ using OWL and our Relational.OWL ontology, we can regard this representation itself as a novel ontology. With this tailored ontology-based representation of the database schema, we are able to represent the data stored in that specific database. As a result, data stored in a relational database can be represented as instances of its own OWL schema.

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 RDF and OWL, know the Relational.OWL ontology (or a semantic equivalent), and have access to the OWL schema representation $\mathcal{S}_i$ of the database $\mathcal{RDB}_i$. As we have mentioned above, the Relational.OWL representation $\mathcal{ROWL}_i$ of a relational database $\mathcal{RDB}_i$ consists of two parts, the schema ontology $\mathcal{S}_i$ and its corresponding data instances $\mathcal{I}_i$.

Using the schema $\mathcal{S}_i$ as a novel ontology means to represent the data stored in the database $\mathcal{RDB}_i$ using a tailored data representation technique. As a result, the data can be handled using common RDF/OWL techniques for data backups, data exchanges, or any kind of data processing tasks within the Semantic Web. A sample data set of a relational database is provided in Fig. 2.

A summary of all the classes and relationships among them, together with a complete database representation can be found in [19].

3.2 Querying RDF Data

Despite the possibility to query RDF graphs in their XML representation using XML query languages like XQuery [9], their possibilities to query the graph for matching triples is rather rudimentary, ignoring the kind of information, which

```
<...>
<db:ADDRESS>
 <db:ADDRESS.ADDRESSID>6824</db:ADDRESS.ADDRESSID>
 <db:ADDRESS.STREET>Campus de Arrosadia</db:ADDRESS.STREET>
 <db:ADDRESS.ZIP>31006</db:ADDRESS.ZIP>
 <db:ADDRESS.CITY>Pamplona</db:ADDRESS.CITY>
 <db:ADDRESS.COUNTRYID>152</db:ADDRESS.COUNTRYID>
</db:ADDRESS>
<db:COUNTRY>
 <db:COUNTRY.COUNTRYID>152</db:COUNTRY.COUNTRYID>
 <db:COUNTRY.NAME>España</db:COUNTRY.NAME>
</db:COUNTRY>
</...>
```

Fig. 2. Data Representation

can be revealed using reasoning mechanisms. In fact, all queries have to be expressed as if they were treating *real* XML documents and not RDF graphs.
Hence, it soon became obvious, that tailored query languages for the Semantic Web languages (e.g. RDF) were required. Naturally, most languages were based on the SQL syntax in order to be easily understood and adopted by a broad community. As we have shown in [18], these early languages like RDQL [24] have one major drawback: they are not closed, i.e. the results of such queries are not valid RDF triples, but a list of possible variable bindings. Hence, the query results cannot be processed using ordinary reasoning mechanisms of normal Semantic Web applications.
An RDF query language has to fulfill one main characteristic for being able to describe a mapping between a relational database and the Semantic Web using our Relational.OWL [19] technique: it has to be closed. Otherwise, the queries used within the mapping process would not return valid RDF graphs but simple variable bindings. Having chosen a closed query language, the expressiveness of a mapping only depends on the query language itself.
In this paper we use the upcoming query language SPARQL [22] as an RDF query language representative, since it will hopefully be recommended soon as a de facto standard by the W3C. SPARQL is an extension of RDQL, eliminating many of its drawbacks, like lack of expressiveness and completeness [18]. Despite its novelty, the Jena Framework [2] already supports SPARQL using its ARQ extension.
Indeed, we have shown in [20], that the combination of SPARQL and Relational.OWL could replace existing interfaces for the access of relational databases out of the Semantic Web. We successfully simulated the basic operations $\{\sigma, \pi, \cup, -, \times\}$ of the relational algebra and showed how to express a join operation with SPARQL. As we will see below, it can easily be deduced from the cartesian product - just as it is done within the relational algebra.
Consider a sample database, which contains personal and contact information of e.g. business partners and contains the following two relations:

`Address(AddressID, Street, ZIP, City, CountryID)` and
`Country(CountryID, Name)`.

A typical join operation between these two relations could be as follows:

$$\sigma_{Address.CountryID=Country.CountryID}(r(Address) \times r(Country)).$$

Since SPARQL does not provide the possibility to specify nested queries, we have to express the (equi-)join operation using a combination of a cartesian product and two selections (cf. [20]). A possible SPARQL query, which holds the same constraints like the relational algebra expression above and thus may be regarded as a correspondent to the (equi-)join operation of the relational algebra can be found in Fig. 3. We again refer to [20] for a detailed and complete analysis of the basic relational operation equivalents in SPARQL.

```
PREFIX    rdf:[...]
PREFIX    db :[...]
CONSTRUCT {?a ?b ?c;
               ?e ?f}
WHERE     {{?a ?b ?c;
                rdf:type db:ADDRESS} .
           {?d ?e ?f;
                rdf:type db:COUNTRY} .
           {?a db:ADDRESS.COUNTRYID ?x} .
           {?d db:COUNTRY.COUNTRYID ?x}}
```

Fig. 3. Sample SPARQL Query

Taking into account the possibility to query the legacy data formerly stored in relational databases using a query language like SPARQL, we have achieved a reasonable alternative for Semantic Web applications to access such relational data. As a consequence, all kinds of legacy data stored in relational databases become an integral part of the Semantic Web.

4 Relational to Semantic Mapping

Despite being processable by any application understanding RDF, the data extracted using Relational.OWL still lacks *real* semantic meaning. Indeed, the information originally stored in relational tables is represented within a table object and not within an appropriate Semantic Web object, e.g. an `http://www.w3.org/2000/10/swap/pim/contact#Person` object. This drawback has to be accepted in order to achieve an automatic transformation from relational databases to the Semantic world.

Nevertheless, many applications still require the data to be represented as *real* semantic objects, for being able to perform reasoning tasks or further data processing. To meet the demands of such applications, a data mapping from the relational to the required data representation is needed.

4.1 Requirements

Common approaches like [8] introduce a special mapping language, which has to be understood and adopted by all administrators needing to perform a single mapping from a relational database to the Semantic Web. Our technique goes one step further and uses common RDF query languages for the mapping task. The following requirements have to be kept, for being able to use such query languages as a mapping language.

Relational.OWL: Contrary to a common mapping, where the relational data is directly translated into the Semantic Web, our approach passes one additional step. First, we represent the data stored in the original relational database in a semantic-rich format, i.e. in RDF. This step is either done exporting the complete data and schema sets into RDF using the Relational.OWL application [4] or using the virtual database representation provided by RDQuery [3]. Please note, that both data transformations methods are processed automatically without any human intervention. Both techniques result in a Semantic Web representation of the data and schema components of the original relational database.

Closed Query Language: We are potentially able to perform a mapping using any of the upcoming query languages, as long as it is closed within RDF and contains a construct similar to the `CONSTRUCT` clause in SPARQL (cf. [22]). Otherwise the resulting variable bindings would have to be translated again into RDF. We have chosen SPARQL as a representative query language, since it is easy to understand, its syntax is based on SQL, it is as powerful as the relational algebra in its expressiveness (cf. [20]), and will hopefully soon be recommended as a de facto standard by the W3C.

Target Ontology: Although the Relational.OWL representation (cf. Section 4.2) of the database is processable by virtually any Semantic Web application, it still lacks *real* semantics, since the data is represented as it was stored in the relational database, i.e. stored in tables and columns. Since we want to assign this data a real meaning, we require a target ontology, it can be mapped to.

4.2 Definitions

In this section we define the basic terms used for our relational database to Semantic Web mapping approach. First, we introduce the *semantic translation* of relational databases into the Semantic Web:

Definition 1 (Semantic Translation). *The semantic translation* $\mathcal{ST}(\mathcal{RDB}, \mathcal{ROWL})$ *of a relational database* $\mathcal{RDB}$ *into its Relational.OWL representation* $\mathcal{ROWL}$ *(see Definition 2 below) is an automatic translation process, where for each* $\mathcal{RDB}$ *and its schema components, a Semantic Web correspondent is created (cf. Section 3.1).*

In this context, the *Relational.OWL representation* of a database is:

Definition 2 (Relational.OWL Representation). *The Relational.OWL representation of a relational database is described by $\mathcal{ROWL}(\mathcal{S},\mathcal{I})$, where $\mathcal{S}$ is the schema representation of the database as seen in Section 3.1 and $\mathcal{I}$ contains the corresponding data instances of the schema components described with $\mathcal{S}$.*

Having created a Relational.OWL representation of the corresponding database, we are now able to perform a *mapping*:

Definition 3 (Mapping). *A mapping $\mathcal{M}$ from a relational database to the Semantic Web is a four-tuple $\mathcal{M}(\mathcal{RDB},\mathcal{ROWL},\mathcal{TO},\mathcal{Q})$, with $\mathcal{RDB}$ being the source database, $\mathcal{ROWL}$ the Relational.OWL representation of $\mathcal{RDB}$, $\mathcal{TO}$ the target ontology, and $\mathcal{Q}$ the mapping query, expressed in a (closed) query language $\mathcal{QL}$.*

Contrary to the *Relational.OWL representation* created with an automatic semantic translation, a *mapping* has to be stated manually using a query Q. The mapping is correct, iff querying $\mathcal{ROWL}$ with $\mathcal{Q}$ results in one ore multiple instances of $\mathcal{TO}$. Hence $\mathcal{Q}$ has to fulfill two main properties. First, it has to be adequate in regard to $\mathcal{ROWL}$, i.e. return the desired result and secondly, the result has to be formatted as instances of $\mathcal{TO}$.

4.3 Mapping Process

The complete relational data to RDF mapping process is illustrated in Fig. 4. It consists of two main steps, which were already introduced in the previous sections.
First, the Relational.OWL representation of the schema and the data components of the original data source are generated. The schema representation becomes thereby an instance of the Relational.OWL ontology. In turn, the data items converted become instances of the schema ontology just created. This step could either be performed using the Relational.OWL application [4], i.e. the schema and data components are translated statically in a one-time process, or using a virtual representation of that RDF model, e.g. with RDQuery [3]. The advantage of the latter is obvious, since the data stock, on which the queries are performed, is always up-to-date. This cannot be guaranteed using the Relational.OWL application. Nevertheless, if the source database does not change frequently, a static translation into the Relational.OWL representation could be enough.
Having created the Relational.OWL representation of the relational database, the second step including the actual mapping can be performed.
The RDF model just created may now be queried with an arbitrary RDF query language. As long as the query language is closed, the resulting query response is again within the Semantic Web, i.e. it is a valid RDF model or graph and may then be processed by other Semantic Web applications using their own built-in functionality for reasoning tasks.

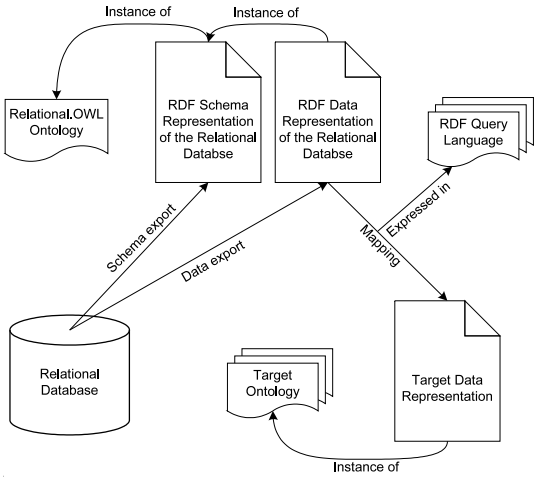

Fig. 4. Mapping Process

Using the `CONSTRUCT` clause of a query language like SPARQL (cf. [22]), the resulting data items can be inserted into an arbitrary RDF skeleton. This property of the query language is vaguely comparable to an XSLT-Stylesheet [1]. If we specify an adequate RDF skeleton, we can achieve the resulting RDF model to correspond to an instance of the intended target ontology. The RDF skeleton in the `CONSTRUCT` clause of the SPARQL query becomes hereby the pivotal part of the actual mapping process. A sample mapping query is provided in Section 4.4.

4.4 Sample Mapping

In this section we present a sample relational data to RDF/OWL mapping using SPARQL as our chosen mapping language, since it fulfills all of our requirements and will hopefully be recommended as a de facto standard by the W3C soon. Despite its novelty, SPARQL is already supported by the Jena Framework [2].
Consider a Semantic Web application developer, who requires access to the data stored in the database introduced in Section 3.2. Since he assumes the database schema to be quite stable, he decides to create a mapping from the relational data model to Semantic Web objets based on the `vCard` ontology [12]. A possible mapping query, which gives Semantic Web applications the possibility to access the data using its own built-in functionality and enables them to perform common reasoning operations is given in Fig. 5.
After specifying the prefix definitions for `vCard`, `rdf`, and `db` in the `PREFIX` clause, the skeleton of the resulting RDF objects is defined in the `CONSTRUCT` part of the query. At first, a new anonymous node of type `vCard:ADR` is created. This object contains the attributes `vCard:Street`, `vCard:Locality`, `vCard:Pcode`, and `vCard:Country` and could easily be extended by further attributes either specified in the `vCard` ontology or in other RDF-Schema files. The values cor-

```
PREFIX vCard:<http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX db:   <http://www.dbs.cs.uni-duesseldorf.de/RDF/address_schema.owl#>
CONSTRUCT {_:v rdf:type vCard:ADR;
               vCard:Street ?street;
               vCard:Locality ?locality;
               vCard:Pcode ?pcode;
               vCard:Country ?country}
WHERE {{?a rdf:type db:ADDRESS;
           db:ADDRESS.ZIP ?pcode;
           db:ADDRESS.STREET ?street;
           db:ADDRESS.CITY ?locality;
           db:ADDRESS.COUNTRYID ?x}.
       {?d rdf:type db:COUNTRY;
           db:COUNTRY.NAME ?country;
           db:COUNTRY.COUNTRYID ?x}}
```

Fig. 5. Sample Mapping Query

responding to the given attributes are specified by free variables, bound in the following `WHERE` clause.

The actual linkage to the original database is performed in the `WHERE` clause of the SPARQL query, i.e. each of the free variables specified in the `CONSTRUCT` clause is bound to a column of the original database. To be more precise, the attributes are bound to the data instances $\mathcal{I}$ of the RDF representation $\mathcal{ROWL}$ of the relational database. Please note, that the mapping specification is identical for a *virtual* RDF model like in RDQuery [3] or a *static* data representation, e.g. with the Relational.OWL application [4]. Being stored on two different tables (`ADDRESS` and `COUNTRY`), the required data is joined using the `?x` variable (cf. [20]).

Having created a suitable mapping from the Relational.OWL representation of the database to the target `vCard` ontology, the resulting data can be processed by any Semantic Web application as usual. In Fig. 6 a sample result set of the mapping query provided in Fig. 5 is given.

4.5 Characteristics

The major characteristics of our relational database to RDF/OWL mapping approach are discussed in this section.

Combination of Automatic and Manual Mappings: The mapping approach presented in this paper is suitable for most relational database scenarios. If we have to handle with constantly changing database schemas, an automatic mapping with Relational.OWL into the Semantic Web is the best choice. Indeed, an automatic mapping with Relational.OWL does not add real semantics to the RDF objects, but at least, the data is processable by any Semantic Web application without having to update the mapping every time the schema changes.

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#" >
 <rdf:Description rdf:nodeID="A0">
     <vCard:Pcode>40225</vCard:Pcode>
     <rdf:type rdf:resource="http://www.w3.org/2001/vcard-rdf/3.0#ADR"/>
     <vCard:Street>Universitätsstr. 1</vCard:Street>
     <vCard:Locality>Düsseldorf</vCard:Locality>
     <vCard:Country>Deutschland</vCard:Country>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A1">
     <vCard:Pcode>31006</vCard:Pcode>
     <rdf:type rdf:resource="http://www.w3.org/2001/vcard-rdf/3.0#ADR"/>
     <vCard:Street>Campus de Arrosadia</vCard:Street>
     <vCard:Locality>Pamplona</vCard:Locality>
     <vCard:Country>España</vCard:Country>
 </rdf:Description>
</rdf:RDF>
```

Fig. 6. Sample Mapping Result

In many application areas, the risk of having to update the mapping is either negligible or consciously taken into account, since data with real semantics is required. For these cases, an additional, manual mapping from the Relational.OWL representation to a target ontology would be appropriate. This may easily be done using a suitable query language. Please note, that all present relational database to Semantic Web approaches require the mapping to be updated, whenever the schema of the database changes, whereas our technique provides an automatic fallback for such situations.

Mapping within the Semantic Web: The complete mapping process from the relational database to RDF objects with real semantics is performed using Semantic Web applications. As a result, two different mapping architectures are possible. The first and most reliable possibility is, that such mappings are processed by small wrapper applications providing the Semantic Web applications with the required target data. Taking place within the Semantic Web, the applications may nevertheless opt to create the mapping by themselves using their own built-in functionality.

Well-known Mapping Language(s): One of the main advantages of our approach is, that it does not require a new mapping language to be adopted, since it is completely based on current RDF/OWL-techniques. Contrary to approaches like [7], Semantic Web application developers needing access to data actually stored in relational databases do not have to learn yet another mapping language, but are able to use their preferred RDF query language, as long as it fulfills the requirements mentioned in Section 4.1.

5 Evaluation

We have evaluated the performance of our relational database to Semantic Web approach using RDQuery [3]. It is a wrapper system, which enables Semantic Web applications to access and query data actually stored in relational databases using their own built-in functionality. RDQuery automatically translates SPARQL and RDQL queries into SQL and is thus able to perform the relational to semantic mapping in one step. Providing an adequate mapping query, the query is translated into SQL, the underlying relational database is queried, and the results are returned in the required format. RDQuery is thereby able to recognize the basic operations of the relational algebra within the SPARQL query (cf. [20]) and to translate them into SQL. Hence, most of the workload, including join and projection operations, is not processed directly by RDQuery, but passed to the underlying database with the generated SQL query.

Following the example shown in Section 4.4, we have created 18 different queries which map relational data to the vCard ontology (cf. http://www.w3.org/TR/vcard-rdf). We have categorized these queries into three different classes, depending on their complexity referring to the relational algebra (i.e. `selection`, `projection`, and `join`). Each of the categories contains six of the queries. We first measured the time required by RDQuery to translate the queries into SQL and then the time passed for the complete mapping process, including query translation, query execution via JDBC using a MySQL database, and the data translation back into RDF. The database, the queries were tested on is based on the `northwind` database and contains eight tables with a total of about 3000 tuples. Unlike the first measurement, the second depends on various factors, like network or database performance, which can hardly be influenced by RDQuery. In Table 1, the average execution time of the query translations and mapping processes for each mapping category is given.

	Execution time [s]	
SPARQL Query	Query Translation	mapping process
Selection	0.020	0.050
Projection	0.018	0.052
Join	0.023	0.067

Table 1. Average Execution Time for a SPARQL Mapping

The performance results show, that the execution time of both, the query translations and the complete mapping process are barely measurable, lying most of them far below 100 milliseconds. Even the more complex join operations were translated and executed at an average of 67 milliseconds. Consequently, our mapping relational data to Semantic Web approach enables applications to access legacy data stored in relational databases in real-time, as if that data would actually be part of the Semantic Web.

6 Discussion and Future Work

In this paper we have described how to map data from relational databases into a real RDF representation using a Semantic Web query language. To use such query languages for a mapping purpose, three main requirements have to be met. First, the relational database (i.e. its schema and data components) has to be described using the Relational.OWL ontology. This automatic semantic representation of the relational database can then be queried using any RDF query language. If the adopted query language is closed, the resulting RDF graph can be specified to match the the target ontology, the original database shall be mapped to.
The approach presented in this paper is based on mapping the Relational.OWL representation of relational databases manually into Semantic Web objects with real semantics. We are thus planning to analyze, whether existing (semi-)automatic schema and ontology matching approaches (cf. [10, 23]) could provide reasonable results in matching an existing relational schema to a target ontology. The expressiveness within the mapping process depends directly from the query language used, i.e. a more complex mapping cannot be stated with an elementary query language. For instance, we showed in [20], that all the basic operations of the relational algebra can be expressed with SPARQL. Nevertheless, its has some considerable limitations, since it does not support aggregations or nested queries. A further restriction concerns data manipulation or data updates, which is still not supported by most RDF query languages. We are currently analyzing, whether SPARQL could be extended to support such operations for enabling Semantic Web applications to manipulate the data actually stored on the relational database.

References

1. XSL Transformations (XSLT). http://www.w3.org/TR/1999/REC-xslt-19991116, 1999.
2. Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net/, 2006.
3. RDQuery. http://sourceforge.net/projects/rdquery/, 2006.
4. Relational.OWL. http://sourceforge.net/projects/relational-owl/, 2006.
5. Yuan An, Alexander Borgida, and John Mylopoulos. Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences. In *CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Cyprus, Part II*, volume 3761 of *LNCS*, pages 1152–1169. Springer, 2005.
6. Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.
7. Christian Bizer. D2R MAP-A Database to RDF Mapping Language. In *WWW2003, The Twelfth International World Wide Web Conference*, Budapest, Hungary, 2003. poster presentation.
8. Christian Bizer and Andy Seaborne. D2RQ -Treating Non-RDF Databases as Virtual RDF Graphs. In *The Semantic Web - ISWC 2004: Third International Semantic Web Conference*, Hiroshima, Japan, 2004. poster presentation.

9. Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/2005/CR-xquery-20051103/, 2005. W3C Candidate Recommendation.
10. AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Ontology Matching: A Machine Learning Approach. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 385–404. Springer, 2004.
11. Stephen Harris and Nigel Shadbolt. SPARQL Query Processing with Conventional Relational Database Systems. In *Web Information Systems Engineering - WISE 2005 Workshops, New York, NY, USA, Proceedings*, volume 3807 of *Lecture Notes in Computer Science*, pages 235–244. Springer, 2005.
12. Renato Iannella. Representing vCard Objects in RDF/XML. http://www.w3.org/TR/vcard-rdf, 2001. W3C Note.
13. Gregory Karvounarakis, Vassilis Christophides, Dimitris Plexousakis, and Sofia Alexaki. Querying RDF Descriptions for Community Web Portals. In *17èmes Journées Bases de Données Avancées, BDA'2001, Agadir, Maroc*, pages 133–144, 2001.
14. Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/, 2004.
15. Sergey Melnik. Storing RDF in a Relational Database. http://www-db.stanford.edu/ melnik/rdf/db.html, 2001.
16. Zhengxiang Pan and Jeff Heflin. DLDB: Extending Relational Databases to Support Semantic Web Queries. In *PSSS1 - Practical and Scalable Semantic Systems, Proceedings of the First International Workshop on Practical and Scalable Semantic Systems*, volume 89 of *CEUR Workshop Proceedings*, 2003.
17. Cristian Pérez de Laborda and Stefan Conrad. 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*, volume 79 of *CEUR*, pages 123–130. RWTH Aachen, 2003.
18. Cristian Pérez de Laborda and Stefan Conrad. Querying Relational Databases with RDQL. In Rainer Eckstein and Robert Tolksdorf, editors, *Berliner XML Tage*, pages 161–172, 2005.
19. Cristian Pérez de Laborda and Stefan Conrad. Relational.OWL - A Data and Schema Representation Format Based on OWL. In *Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005)*, volume 43 of *CRPIT*, pages 89–96, Newcastle, Australia, 2005. ACS.
20. Cristian Pérez de Laborda and Stefan Conrad. Bringing Relational Data into the Semantic Web using SPARQL and Relational.OWL. In *Semantic Web and Databases, Third International Workshop, SWDB 2006, Co-located with ICDE, Atlanta, USA, April 2006*. IEEE Computer Society, 2006.
21. Johan Petrini and Tore Risch. Processing Queries over RDF views of Wrapped Relational Databases. In *1st International Workshop on Wrapper Techniques for Legacy Systems, WRAP 2004, Delft, Holland*, 2004.
22. Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. http://www.w3.org/TR/2006/WD-rdf-sparql-query-20060220/, 2006. W3C Working Draft.
23. Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
24. Andy Seaborne. RDQL - A Query Language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, 2004.