Bringing Relational Data into the Semantic Web using SPARQL and Relational.OWL

Cristian Pérez de Laborda, Stefan Conrad
perezdel@cs.uni-duesseldorf.de

SWDB 2006
Atlanta, GA

April 8, 2006
1 Introduction
   - Relational.OWL
   - Relational Databases and the SW
   - Query Languages for the Semantic Web

2 Relational Databases and RDF-QL
   - SPARQL vs. SQL
   - Selection
   - Projection
   - Set Union
   - Set Difference
   - Cartesian Product
   - (Equi-)Join

3 RDQuery

4 Conclusion and Future Work
Outline

1 Introduction
   - Relational.OWL
   - Relational Databases and the SW
   - Query Languages for the Semantic Web

2 Relational Databases and RDF-QL
   - SPARQL vs. SQL
   - Selection
   - Projection
   - Set Union
   - Set Difference
   - Cartesian Product
   - (Equi-)Join

3 RDQuery

4 Conclusion and Future Work
The Relational.OWL ontology is a central representation of abstract schema components of a relational database.

The schema of any relational database can be described using Semantic Web techniques.

Based on this schema representation, the actual data can be described.
Relational.OWL Ontology:

```xml
<owl:Class rdf:ID="Database">
    <rdfs:label xml:lang="en">Database</rdfs:label>
    <rdfs:subClassOf rdf:resource="&rdf;Bag"/>
</owl:Class>

<owl:Class rdf:ID="Table">
    <rdfs:label xml:lang="en">Table</rdfs:label>
    <rdfs:subClassOf rdf:resource="&rdf;Seq"/>
</owl:Class>

<owl:Class rdf:ID="Column">
    <rdfs:label xml:lang="en">Column</rdfs:label>
    <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasTable">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="#Database"/>
    <rdfs:range rdf:resource="#Table"/>
    <rdfs:label xml:lang="en">hasTable</rdfs:label>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasColumn">
    <rdfs:domain rdf:resource="#Table"/>
    <rdfs:range rdf:resource="#Column"/>
    <rdfs:label xml:lang="en">hasColumn</rdfs:label>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="identifiesBy">
    <rdfs:domain rdf:resource="#Table"/>
    <rdfs:range rdf:resource="#Column"/>
    <rdfs:label xml:lang="en">isIdentifiedBy</rdfs:label>
</owl:ObjectProperty>
```

### Database Schema:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>COUNTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK ADDRESSID</td>
<td>PK COUNTRYID</td>
</tr>
<tr>
<td>STREET</td>
<td>ZIP</td>
</tr>
<tr>
<td>FK1</td>
<td></td>
</tr>
</tbody>
</table>

### Schema Representation:

```
<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"/>
  <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>
```

...
Relational.OWL (IV)

Database Schema:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>COUNTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>ADDRESSID</td>
</tr>
<tr>
<td>STREET</td>
<td>NAME</td>
</tr>
<tr>
<td>ZIP</td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td></td>
</tr>
<tr>
<td>COUNTRYID</td>
<td></td>
</tr>
</tbody>
</table>

| COUNTRY | |
|---------|
| PK | COUNTRYID |

Data Representation:

```
<db:ADDRESS>
  <db:ADDRESS.ADDRESSID>3248</db:ADDRESS.ADDRESSID>
  <db:ADDRESS.STREET>Königsallee 21</db:ADDRESS.STREET>
  <db:ADDRESS.ZIP>40212</db:ADDRESS.ZIP>
  <db:ADDRESS.CITY>Düsseldorf</db:ADDRESS.CITY>
  <db:ADDRESS.COUNTRYID>32</db:ADDRESS.COUNTRYID>
</db:ADDRESS>

<db:ADDRESS>
  <db:ADDRESS.ADDRESSID>6824</db:ADDRESS.ADDRESSID>
  <db:ADDRESS.STREET>Iñigo Arista 1</db:ADDRESS.STREET>
  <db:ADDRESS.ZIP>31007</db:ADDRESS.ZIP>
  <db:ADDRESS.CITY>Pamplona</db:ADDRESS.CITY>
  <db:ADDRESS.COUNTRYID>152</db:ADDRESS.COUNTRYID>
</db:ADDRESS>

<db:COUNTRY>
  <db:COUNTRY.COUNTRYID>32</db:COUNTRY.COUNTRYID>
  <db:COUNTRY.NAME>Deutschland</db:COUNTRY.NAME>
</db:COUNTRY>

<db:COUNTRY>
  <db:COUNTRY.COUNTRYID>152</db:COUNTRY.COUNTRYID>
  <db:COUNTRY.NAME>España</db:COUNTRY.NAME>
</db:COUNTRY>
```
Relational Databases and the SW

Problem:
- The Semantic Web still lacks *real* data.
- Data is usually modeled and stored relationally.
- This data is hardly accessible from the Semantic Web.
- Manual Mapping is time consuming and not always feasible.

Proposal:
- Automatic transformation of relational data into the Semantic Web with Relational.OWL.
- Access to this data representation using query languages of the Semantic Web.
- Relational legacy data becomes an integral part of the Semantic Web.
Relational Databases and the SW (II)

Currently:

Semantic Web Application → Semantic Web Application

SQL with Mapping → SQL with Mapping

Relational Database

Proposal:

Semantic Web Application → Semantic Web Application

RDF-QL → RDF-QL

Relational,OWL Representation

Relational Database
Most query languages for the Semantic Web are still under development.

- Different query languages:
  - RQL, RDQL, SPARQL
  - Metalog, TRIPLE
  - XQuery, Xcerpt

- Could a query language together with Relational.OWL replace current SQL access to legacy databases?

- Problem: Query languages vary in their expressiveness and closeness.
Query Languages for the Semantic Web

Most query languages for the Semantic Web are still under development.

- Different query languages:
  - RQL, RDQL, SPARQL
  - Metalog, TRIPLE
  - XQuery, Xcerpt

- *Could a query language together with Relational.OWL replace current SQL access to legacy databases?*

- Problem: Query languages vary in their expressiveness and closeness.

Reasons for SPARQL

- SPARQL is closed using its `CONSTRUCT` or `DESCRIBE` clauses.
- RDQL is not suitable (cf. [PeCo05b]).
- SPARQL will hopefully be recommended soon by the W3C.
- SPARQL is already supported by Jena using its ARQ extension.
1 Introduction
- Relational.OWL
- Relational Databases and the SW
- Query Languages for the Semantic Web

2 Relational Databases and RDF-QL
- SPARQL vs. SQL
- Selection
- Projection
- Set Union
- Set Difference
- Cartesian Product
- (Equi-)Join

3 RDQuery

4 Conclusion and Future Work
SPARQL vs. SQL

Expressiveness:

- The Relational.OWL representation of a relational databases can be queried using current RDF-QL.
- SPARQL as sample query language.
- How expressive is this combination?
- Direct comparison to SQL unfair
  - basic operations of the relational algebra \{\sigma, \pi, \cup, -, \times\}, \{\bowtie\}.
Selection

Rel. Algebra: \[ \sigma_{\text{Name}='Australia'}(r(\text{Country})) \]

SPARQL:

```
PREFIX rdf:[...]
PREFIX db :[...]
DESCRIBE ?a
WHERE {
  ?a rdf:type db:COUNTRY.
  {?a db:COUNTRY.NAME 'Australia'}
}
```

Result:

```
<rdf:RDF xmlns:j.0="http://.../address_schema.owl#"
         xmlns:rdf="http://.../22-rdf-syntax-ns#">
  <j.0:COUNTRY>
    <j.0:COUNTRY.COUNTRYID>77</j.0:COUNTRY.COUNTRYID>
    <j.0:COUNTRY.NAME>Australia</j.0:COUNTRY.NAME>
  </j.0:COUNTRY>
</rdf:RDF>
```
Projection

Rel. Algebra: $\pi_{\text{Street}, \text{City}}(r(\text{Address}))$

SPARQL:

```sparql
{ ?a rdf:type db:ADDRESS}. FILTER (?b=db:ADDRESS.STREET || ?b=db:ADDRESS.CITY)}
```

Result:

```xml
<rdf:RDF xmlns:j.0="http://.../address_schema.owl#"
  xmlns:rdf="http://.../22-rdf-syntax-ns#">
  <rdf:Description rdf:nodeID="A0">
    <j.0:ADDRESS.CITY>Aachen</j.0:ADDRESS.CITY>
    <j.0:ADDRESS.STREET>Isartorplatz</j.0:ADDRESS.STREET>
  </rdf:Description>
</rdf:RDF>
```
Set Union

Rel. Algebra: \( \pi_{\text{City}}(r(\text{Address})) \cup \pi_{\text{Name}}(r(\text{Country})) \)

SPARQL:

```sparql
PREFIX rdf: [...] PREFIX db: [...] CONSTRUCT 
{ _:v ?b ?c } WHERE 
{ ?a ?b ?c } .
{ ?a db:ADDRESS.CITY ?c } .
{ ?a rdf:type db:ADDRESS } }
UNION
{ ?a db:COUNTRY.NAME ?c } .
{ ?a rdf:type db:COUNTRY } }
```

Result:

```xml
<rdf:RDF xmlns:j.0="http://.../address_schema.owl#"
  xmlns:rdf="http://.../22-rdf-syntax-ns#">
  <rdf:Description rdf:nodeID="A0">
    <j.0:ADDRESS.CITY>Heidelberg</j.0:ADDRESS.CITY>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A1">
    <j.0:COUNTRY.NAME>Ecuador</j.0:COUNTRY.NAME>
  </rdf:Description>
</rdf:RDF>
```
Set Difference

Rel. Algebra: \( \pi_{\text{CountryID}}(r(\text{Country})) - \pi_{\text{CountryID}}(r(\text{Address})) \)

SPARQL:
```sparql
```

Result:
```xml
<rdf:RDF xmlns:j.0="http://.../address_schema.owl#" xmlns:rdf="http://.../22-rdf-syntax-ns#">
  <rdf:Description rdf:nodeID="A8641">
    <j.0:COUNTRY.COUNTRYID>4</j.0:COUNTRY.COUNTRYID>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A8642">
    <j.0:COUNTRY.COUNTRYID>45</j.0:COUNTRY.COUNTRYID>
  </rdf:Description>
  ...
</rdf:RDF>
```
Cartesian Product

Rel. Algebra: \( r(Country) \times r(Address) \)

SPARQL:

```sparql
```

Result:

```xml
<rdf:RDF xmlns:j.0="http://.../address_schema.owl#" xmlns:rdf="http://.../22-rdf-syntax-ns#">
  <rdf:Description rdf:nodeID="A0">
    <j.0:ADDRESS.ADDRESSID>97</j.0:ADDRESS.ADDRESSID>
    <j.0:ADDRESS.STREET>Universitaetsstr. 1</j.0:ADDRESS.STREET>
    <j.0:ADDRESS.ZIP>40225</j.0:ADDRESS.ZIP>
    <j.0:COUNTRY.NAME>New Zealand</j.0:COUNTRY.NAME>
    <j.0:ADDRESS.COUNTRYID>32</j.0:ADDRESS.COUNTRYID>
    <j.0:ADDRESS.CITY>Duesseldorf</j.0:ADDRESS.CITY>
    <rdf:type rdf:resource="http://...schema.owl#ADDRESS"/>
    <rdf:type rdf:resource="http://...schema.owl#COUNTRY"/>
  </rdf:Description>
  ...
</rdf:RDF>
```
(Equi-)Join

Rel. Algebra: \( r(\text{Address}) \bowtie r(\text{Country}) \)

SPARQL:

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

Result:

```xml
<rdf:RDF xmlns:j.0="http://.../address_schema.owl#"
    xmlns:rdf="http://.../22-rdf-syntax-ns#"> 
    <rdf:Description rdf:nodeID="A0"> 
        <j.0:ADDRESS.ADDRESSID>97</j.0:ADDRESS.ADDRESSID> 
        <j.0:ADDRESS.STREET>Universitattsstr. 1</j.0:ADDRESS.STREET> 
        <j.0:COUNTRY.COUNTRYID>32</j.0:COUNTRY.COUNTRYID> 
        <j.0:ADDRESS.ZIP>40225</j.0:ADDRESS.ZIP> 
        <j.0:ADDRESS.COUNTRYID>32</j.0:ADDRESS.COUNTRYID> 
        <j.0:COUNTRY.NAME>Germany</j.0:COUNTRY.NAME> 
        <j.0:ADDRESS.CITY>Duesseldorf</j.0:ADDRESS.CITY> 
        <rdf:type rdf:resource="http://...schema.owl#ADDRESS"/> 
        <rdf:type rdf:resource="http://...schema.owl#COUNTRY"/> 
    </rdf:Description> 
</rdf:RDF>
```
Outline

1 Introduction
   - Relational.OWL
   - Relational Databases and the SW
   - Query Languages for the Semantic Web

2 Relational Databases and RDF-QL
   - SPARQL vs. SQL
   - Selection
   - Projection
   - Set Union
   - Set Difference
   - Cartesian Product
   - (Equi-)Join

3 RDQuery

4 Conclusion and Future Work
RDQuery

Details according to the query

Type your query here

SELECT ?x, ?y, ?z
WHERE (?x, ?y, ?z)
(?x, rdf:type, db:customers)
(?x, db:customers.City, "Berlin")
AND (?y EQ db:customers.ContactName)
USING db for <http://mydb.com#>

Query examples

Database details

Database

categories

customers

customers

CustomerID

CompanyName

ContactName

ContactTitle

Address

City

Translation results

Query Results Query Translation Parser Details RDF Representation

SELECT customers.ContactName
FROM customers
WHERE customers.City = "Berlin"
Outline

1 Introduction
   - Relational.OWL
   - Relational Databases and the SW
   - Query Languages for the Semantic Web

2 Relational Databases and RDF-QL
   - SPARQL vs. SQL
   - Selection
   - Projection
   - Set Union
   - Set Difference
   - Cartesian Product
   - (Equi-)Join

3 RDQuery

4 Conclusion and Future Work
Conclusion and Future Work

Conclusion:

- Relational.OWL opens the relational world to the Semantic Web.
- Semantic Web applications can access relational databases using their own built-in functionality.
- Each legacy database becomes an integral part of the Semantic Web.
- Successfully simulated the basic operations of the relational algebra with SPARQL.
- SPARQL and Relational.OWL are a real alternative to SQL.

Future Work:

- Mapping to a target ontology.
Questions..?

Answers..!