Semantic Web: from Relational DataBase to Ontology

Roberto Casadei
roberto.casadei12@studio.unibo.it

Alma Mater Studiorum – University of Bologna
Master's Degree in Computer science and Engineering
Within the Semantic Web course

1 Introduction

This report describes my final work and research within the course "Semantic Web".
The previous assignments were about

1. introducing the topic of the Semantic Web as well as providing some insights about its spread in the industry
2. modelling a domain of interest and developing a (weak) ontology using RDF Schema
3. modelling a domain of interest and developing a (strong) ontology using OWL

where I chose the "Project Management" domain as the object of my modelling efforts.

This final work is still about the development of an ontology, but the information to be used is (assumed to be already) provided by a Relational DataBase (RDB). In other words, the general problem to be faced is: *how can we convert relational data into RDF/OWL form*?
My goal is twofold:

1. look at the state-of-art of the problem
2. turn the result of such research into action by performing an RDB-to-RDF conversion of a sample database.

2 Basics: Relational DataBases and the Semantic Web

First of all, a natural question may arise: ***why should we care about the mapping between relational databases and RDF***?
The Semantic Web is not a revolution, but a gradual process, a shift towards a Web of Data where information can be more easily accessed, discovered, integrated, and used by both machines and humans. It is conceived as an extension, rather than a replacement, to the "Syntactic Web" where existing tools are reused when possible.
Now, it turns out that, while NoSQL systems and triple stores are increasingly adopted (see Figure 2), **Relational DBMSs still dominate the database market** (see Figure 1). Moreover, if we just consider the Web part of the global data space, because **the majority of dynamic Web content is backed by RDBs** [1], it is clear that the ability to leverage on these data is critical for the success of the Semantic Web.

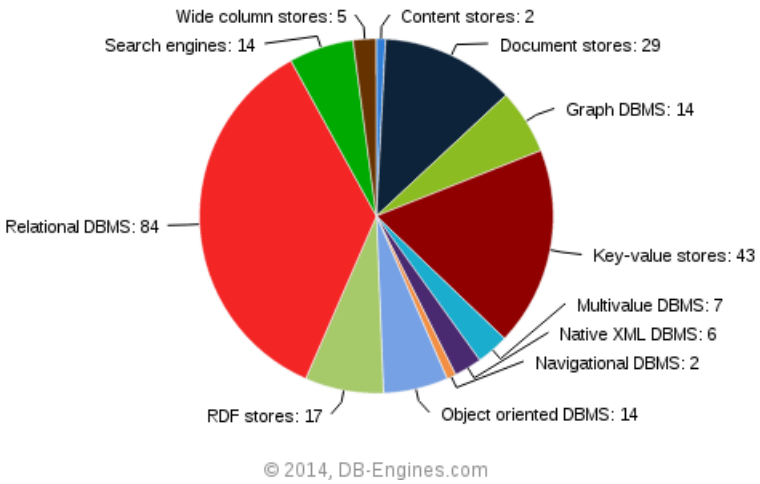

Fig. 1. Number of systems per category, June 2014

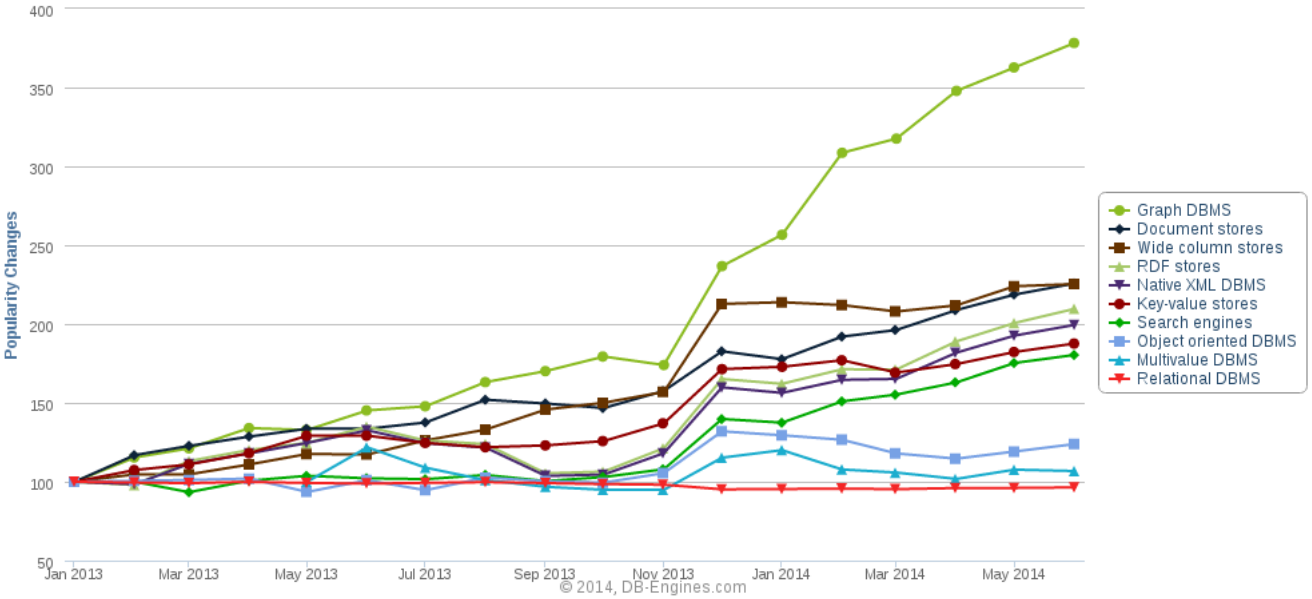

Fig. 2. Popularity changes per category, June 2014

The Semantic Web is the Web of Data and data is everywhere, with relational databases being a widespread container. Thus, **the challenge is to export these data into a common format so to enable them for interoperability within the Semantic Web ecosystem. Such a common format is RDF** (see Figure 3).

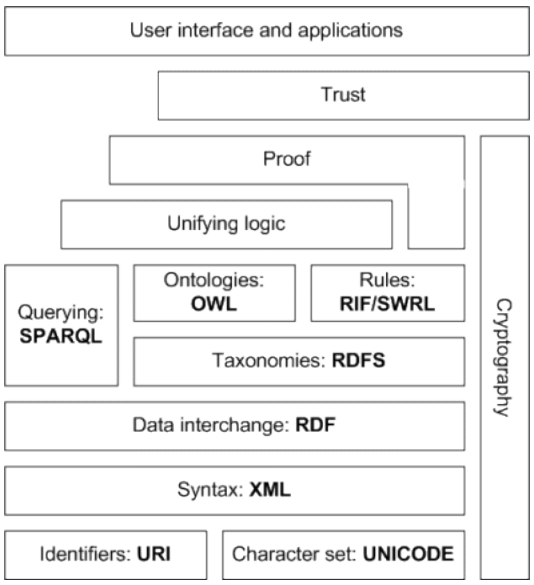

Fig. 3. The Semantic Web stack

It should also be noted that such an interaction between the Semantic Web community and the database community does represent an attractive opportunity for the latter as well, where distribution, integration, and deduction are significant issues.

In a nutshell, the main motivations and opportunities arising from bringing relational databases in the Semantic Web [2] are the following:

- *Integration of RDBs with other data sources* (see above)
- *Generation of a critical mass of data for the Semantic Web*: by automatic extraction of information from RDBs into RDF (see above)
- *Heterogeneous database integration*: with ontology-based integration as an alternative to an integration approach based on conceptual models
- *Semantic annotation of dynamic (RDB-backed) web pages*: augmenting web content with semantic labels which are generated based on the mappings between the RDB schema and existing ontologies
- *Ontology-based data access of the RDB*: allowing users to write semantic queries which are rewritten into SQL
- *Ontology learning*: as ontology development is not an easy task, exploiting the information of an RDB (not just schema and data but also metadata, queries, views, stored procedures) to derive an ontology which can be subsequently extended/refined is alluring
- *Specification of the semantics of relational schemas*: as an alternative to maintaining and keeping aligned the original conceptual model with the relational schema, defining a mapping between the schema and an ontology provides meaning for designers and provides support for DB maintainance and integration as well as representing a common model for database reconciliation

2.1 Converting relational data into RDF data

In general, relational data can be made available in RDF form using two approaches (see Figure 4). One approach consists in performing a *physical* conversion (aka *RDF dump*), where relational data is **extracted, transformed, and loaded (ETL)** into a triple store which represents the ***materialized* view** of the original RDB. The other approach works by providing a ***virtual* view** of the relational database as a RDF graph. In this case, for example, SPARQL queries can be translated into equivalent SQL queries which are then issued against the RDB.

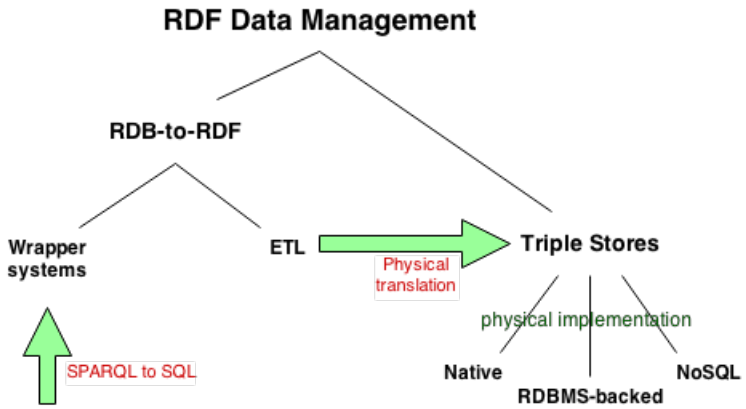

Fig. 4. RDF Data Management and Relational Databases [3]

The creation of an RDF view of relational data allows them to become part of the Web of Data (see Figure 5), thus inheriting significant advantages in terms of access and data integration which make them unleash their value. For example, a typical use case consists in the need of integrating an RDB with other data represented in a different form using RDF as a medium.

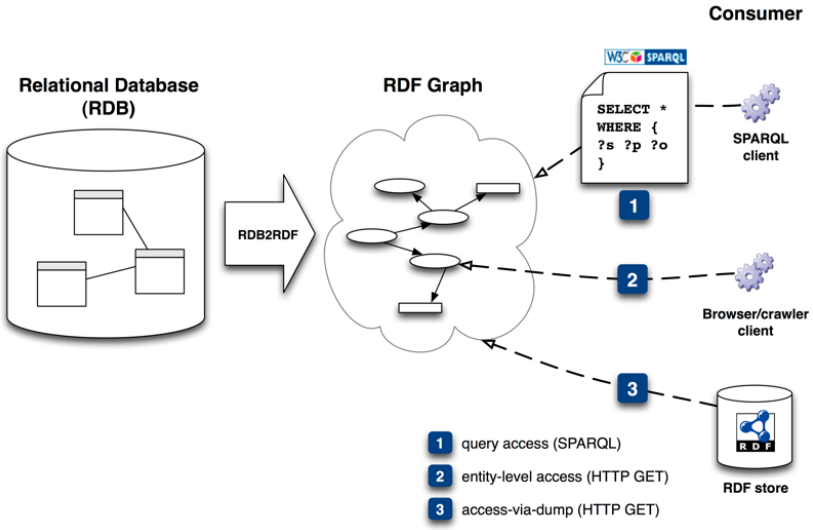

Fig. 5. On the RDF view of a relational database many kinds of access are supported. [4]

3 Mapping between the Relational Model and the RDF Model

It should be noted that the RDB-to-RDF transformation process happens at two levels or stages:

1. at the *intension*-level: where the relational schema is put in relation with a projected or an existing/custom ontology, and
2. at the *extension*-level: where the instances of the relational DB are converted into an RDF graph using the mappings

So, the challenge is to **how to produce the mappings**.
The approach to mapping generation can be

- *Automatic*: where the mappings are automatically generated given the relational schema (and, if possible/needed, some destination ontologies) (see Figure 6)

- *Semi-automatic*: where some mappings are automatically generated but need to be extended or refined by the user (see Figure 7)
- *Manual*: where the user defines the mappings by hand using a RDB-to-RDF mapping language or a tool (see Figure 8)

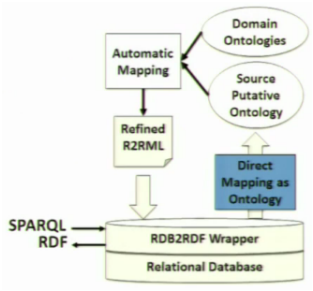

Fig. 6. Automatic mapping. [3]

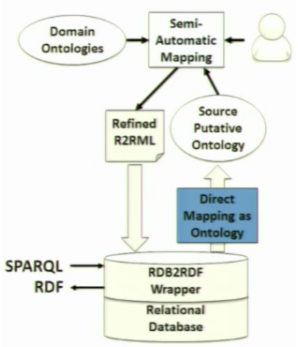

Fig. 7. Semiautomatic mapping. [3]

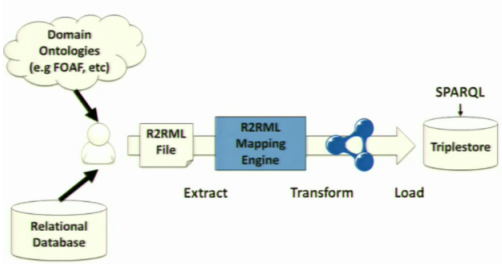

Fig. 8. Manual mapping. [3]

Typically, as ontology languages such as OWL are more expressive then the relational model, the automatically generated mappings are not sufficient to capture a desired level of domain semantics. However, they provide a basis for subsequent refinements. Moreover, rather than simply generating an ontological projection of the relational schema, some existing ontology can be used to improve the quality of mappings or to guide to process within well-defined ontological boundaries. Anyhow, when applications require complex domain semantics to be expressed, the semi-automatic/manual approaches are much probably necessary.

3.1 W3C RDB2RDF activities

In 2008, the RDB2RDF Incubator Group was launched by the W3C with the mission to

> *" to **examine and classify existing approaches** to mapping relational data into RDF and assess whether standardization is possible and/or necessary in this area"*

The survey [5] inspected many projects (prototypes, domain-specific projects, and existing tools) and was carried out by considering the following elements:

- *Creation of mappings*: the approaches are classified in two broad classes
  (a) **Automatic mapping** (aka **local ontology mapping**): where the starting point is the relational schema
  (b) **Domain semantics-driven mapping** (aka **domain ontology mapping**): there is some destination ontology to be populated; usually more precise (also useful for reducing the size of the resulting RDF graph) as more domain semantics is provided
- *Mapping representation and accessibility*: this dimension evaluates the languages used to express the mappings
- *Mapping implementation*: where two implementation strategies – **ETL-based (static)** and **query-based (dynamic)** – are considered
- *Query implementation*: where two possibilities are found: **SPARQL** queries on the resulting RDF repository and SPARQL queries translated to SQL queries (**SPARQL-to-SQL**) on the original RDB
- *Application domain*: the approaches to (and the tools for) mapping generation can be **generic** or **domain-specific**
- *Data integration*: this dimension considers the different mapping methods with respect to data integration

One of the conclusions of such a survey is the assessment of the **need of a standard language** for the representation of RDB-to-RDF mappings. The reason is essentially to allow for the **reuse** of the mappings across different databases and RDBMSs.

With such an input, in 2009, the W3C started an RDB2RDF Working group.

*"The mission of the RDB2RDF Working Group, part of the Semantic Web Activity, is to **standardize languages** for mapping relational data and relational database schemas into RDF and OWL. The two languages are the Direct Mapping (DM) and the RDB2RDF Mapping Language (R2RML)"*

From the analysis of some use cases, a core set of requirements for the R2RML mapping language were proposed [4]:

- Direct mapping support (to produce a local ontology)
- Transformative mapping support (towards a destination ontology)
- Generation of globally unique identifiers
- Sufficient expressivity to support query translation (e.g., SPARQL to SQL)
- Sufficient expressivity to support ETL (RDF dump)
- Datatypes
  - Mapping relational data types to XML Schema data types
  - Extensibility mechanisms to support mappings for vendor-specific data types
- ...Other more technical requirements...

4 Standard RDB-to-RDF mapping languages

4.1 Direct Mapping

The direct mapping produces an RDF representation (also known as the **direct graph**) of a relational database.
The rules for the direct mapping are the following ones [6]:

- **Row type triple**
  - Subject: *row node*
  - Predicate: *rdf:type*
  - Object: *table resource*
- **Literal triple**
  - Subject: *row node*
  - Predicate: *column resource*
  - Object: *literal value*
- **Reference triple**
  - Subject: *row node*
  - Predicate: *foreign key column resource*
  - Object: *(referenced-)row node*

The identifiers are created using the following schema:

- Row node: <**BaseIRI/DB/TableName/PKAttr1=value1;PKAttr2=value2;...**>
  - NOTE: if the table has no primary key, a new blank node is created
- Table resource: <**BaseIRI/DB/TableName**>
- Column resource: <**BaseIRI/DB/TableName#ColumnName**>
- Foreign key column resource: <**BaseIRI/DB/TableName#ref-FKColumn1;FKColumn2;...**>

Note how relative IRIs are concatenated to a base IRI in order to support the uniqueness of identifiers.

The direct mapping suffers of some limitations. For example, as the direct graph reflects the structure and the names of the relational schema, if the schema is particularly complex or not normalized, the result may be poor. Similarly, the IRIs may NOT be user-friendly. Moreover, the relational schema is NOT mapped into an OWL ontology.

4.2 RDB to RDF Mapping Language (R2RML)

R2RML [7] is a standard language for defining mappings to transform an RDB into RDF. An R2RML mapping is itself represented as RDF.

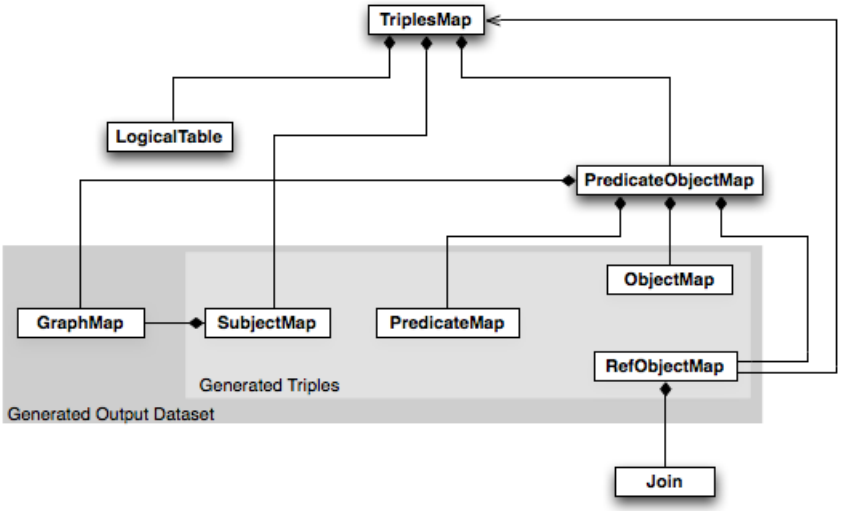

Fig. 9. R2RML overview

R2RML works by mapping **triples maps** to logical tables. A **logical table** is a table, a view, or a SQL query (aka R2RML view). A triple map creates a correspondence between a table row and a set of RDF triples. A triple map is composed of subject maps, predicate maps, and object maps. The **output dataset** consists of a default (unnamed) graph and possibly more (IRI-)named graphs. The generated triples can be assigned to one or more of these graphs.

5 From Relational DataBase to Ontology

Many of the things we would like to do with a relational database (see opportunities and motivations in Section 2) are not limited to the generation of an RDF representation of the RDB content but also need to deal with semantics at higher levels. So, the ***DB to Ontology mapping problem (DB2O)*** is to be tackled.
This problem has been referred to when considering the following distinct but closely related issues [2]:

1. Generation of a brand-new ontology from an RDB
2. Generating an RDF graph which reflects the content of an RDB where RDF statements are annotated in terms of existing ontologies (problem essentially solved by the R2RML standard, see Section 3)
3. Performing semantic queries (e.g., SPARQL queries) on an RDB
4. Discovering the mappings between an RDB and existing ontologies

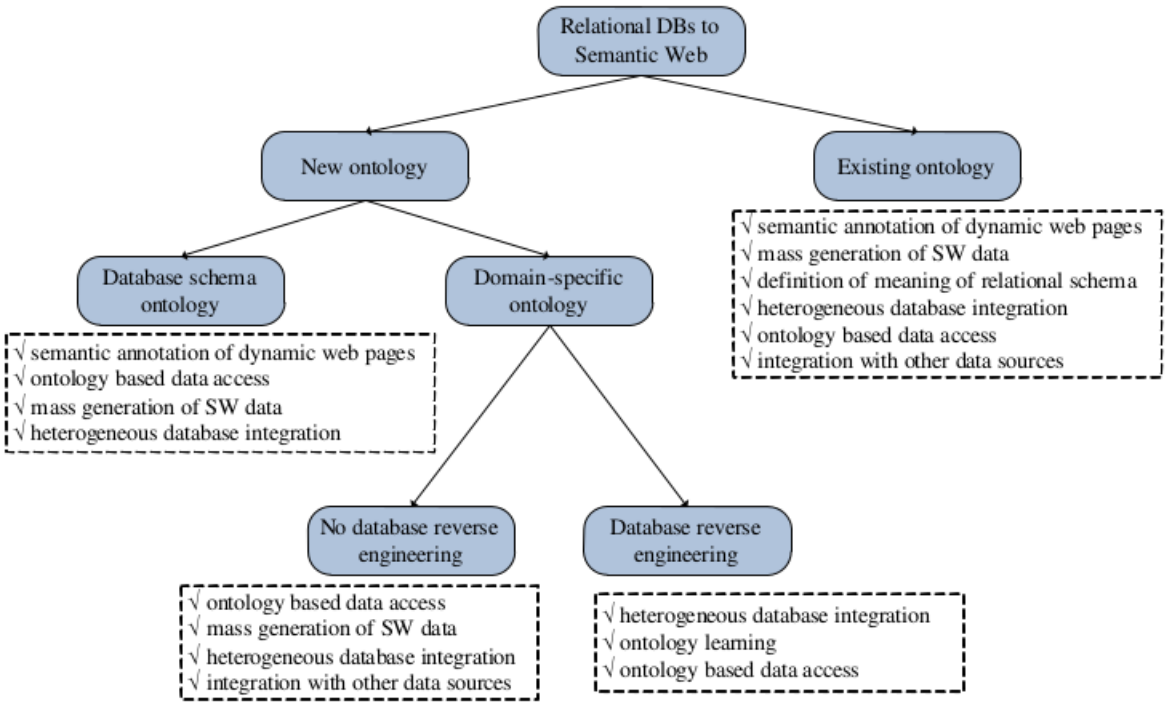

Fig. 10. A total, exclusive classification of RDB-to-SW approaches [2]

A taxonomy of the solutions for the DB2O problem from an ontological perspective can be found in [2] (see Figure 10):

- Level 1: "New ontology" vs. "Existing ontology"
  - Genus: a set of RDB-to-ontology mappings are produced
  - Differentia: a new ontology is created vs. the mappings are defined in terms of an existing ontology
- Level 2: "DB schema ontology" vs. "Domain-specific ontology"
  - Genus: a new ontology is generated
  - Differentia: the new ontology directly reflects the DB schema (*database schema ontology*) vs. the new ontology models domain concepts
- Level 3: "DB reverse engineering" vs. "No DB reverse engineering"
  - Genus: a new domain-specific ontology is produced as output
  - Differentia: a reverse engineering process is vs. not performed in order to extract a conceptual schema which is then mapped to an ontology

In general, the process of extracting an ontology from a relational database is performed by a **mapping engine** (see Figure 11) which uses a set of mappings and rules which may be written by a developer (possibly with a tool) or pre-defined inside the engine.

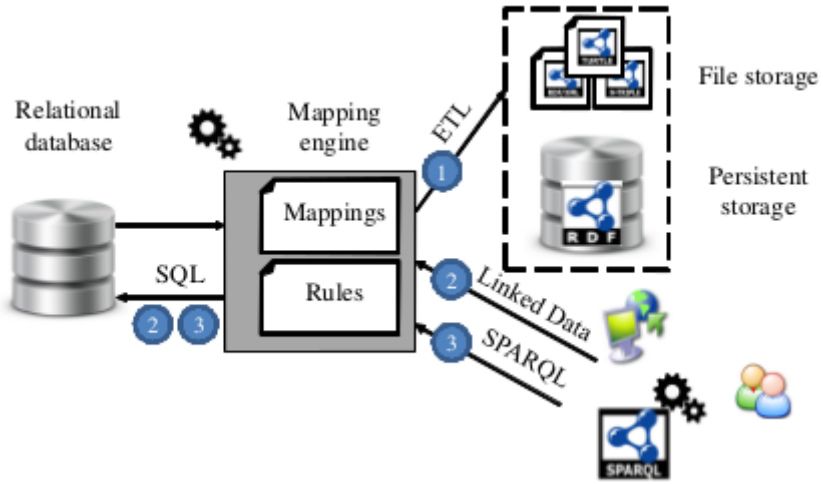

Fig. 11. RDB-to-Ontology [2]

As you may have noted, conceptually there is no difference with respect to the RDB-to-RDF process we have seen in Section 3.

In this section, the focus is on the generation of a brand-new domain-specific ontology from an RDB. For the purpose, it may be useful to differentiate between two kinds of components in ontologies:

- **TBox** (**T**erminological): axioms and definitions about concepts and properties
- **ABox** (**A**ssertion): statements about individuals

Roughly, we would like to generate the TBox (ontology schema) from the DB's intensional knowledge and the ABox (ontology instance data) from the DB's extensional knowledge as depicted in Figure 12.

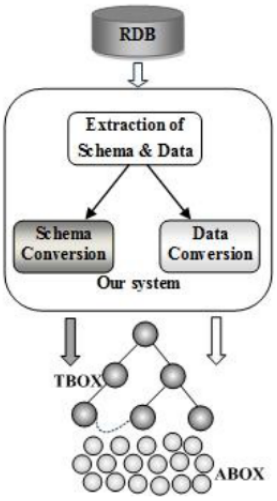

Fig. 12. Ontology TBox and ABox generation [8]

However, note that not just the relational schema but also the instance data can be exploited to derive TBox statements, e.g., by employing correlation analysis and data mining techniques.

5.1 A simple method

A basic method consists in mapping

- tables to classes

- rows to individuals
- columns to data properties
- foreign key columns to object properties
- rows' attributes to property values (or other individuals in case of reference)

(where the OWL terminology is used).
This is sometimes called **"table-to-class, column-to-predicate"** and is somehow similar to what we have seen when covering the RDB-2-RDF Direct Mapping (see Section 4.1).

The plain basic method presented above suffers of many drawbacks (e.g., it would create a class for many-to-many tables as well), however it can be extended and refined into more effective methods/tools which provide some degree of support for features such as:

- Mapping customization
- Conditional mappings
- DB semantics extraction mechanisms
- Pattern-based customization of identifiers (IRIs)
- Application of transformation functions to values
- Mapping attribute domains to classes
- and more

5.2 Intermediate models

Getting an OWL ontology from a RDB can be seen as a **Model-to-Model (M2M)** transformation. The impedance mismatch may result into complex mappings or operational issues. So, a simplification technique consists in splitting the transformation process into two or more phases. Examples can be found in [9] (see Figure 13) and in [10] (in which case the relational schema is converted into an XML Schema which is then transformed into OWL via XSLT); methods for transforming Entity/Relationships (E/R) models into OWL ontologies have also been explored in the literature.

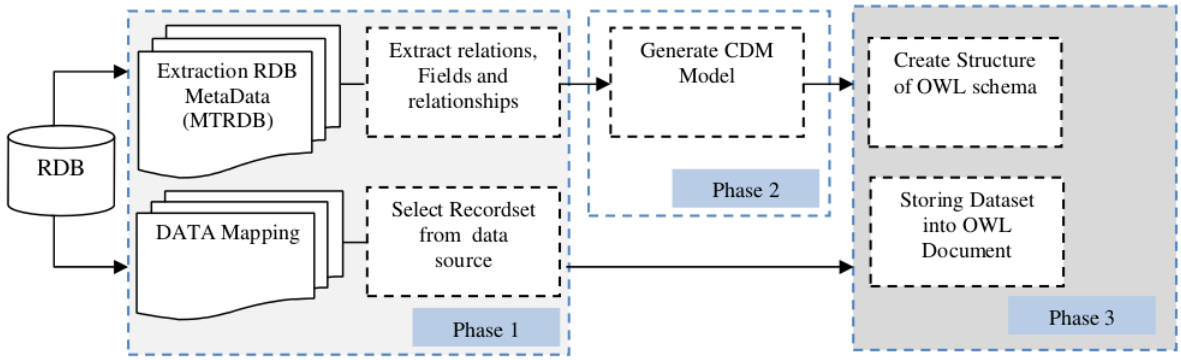

Fig. 13. A Canonical Data Model as an intermediate model.

Often, these approaches make use of **heuristic rules** which looks for characteristics in the relational schema in order to decide how the elements are to be treated. For example, database tables may be distinguished into different kinds based on the number of primary keys or foreign keys. A rough categorization of these rules is provided in [2]. The main issues are the following:

- Efficiency (especially when formal methods are employed)
- Correctness of the inferred meaning
- Expressivity with respect to high-level axioms
- Ontology features to be generated vs. decidability of the resulting ontology

Thus, sometimes the intervention of human domain experts is required at certain decision points, for adjustments, or for validation.

6 Generating an ontology from a sample "Project Management" DB

The goal of this section is to illustrate how to produce an OWL ontology in the domain of project management by extracting the semantics from a relational database.

6.1 The methodology

The process (represented as a black box in Figure 14) has the following inputs:

1. A relational database: schema and instance data
2. Knowledge of the target domain (project management domain)
3. Knowledge of a target ontology language (such as OWL)
4. Knowledge of an RDB-to-ontology mapping language (such as R2RML)

and the output is an (OWL) ontology in the target domain.

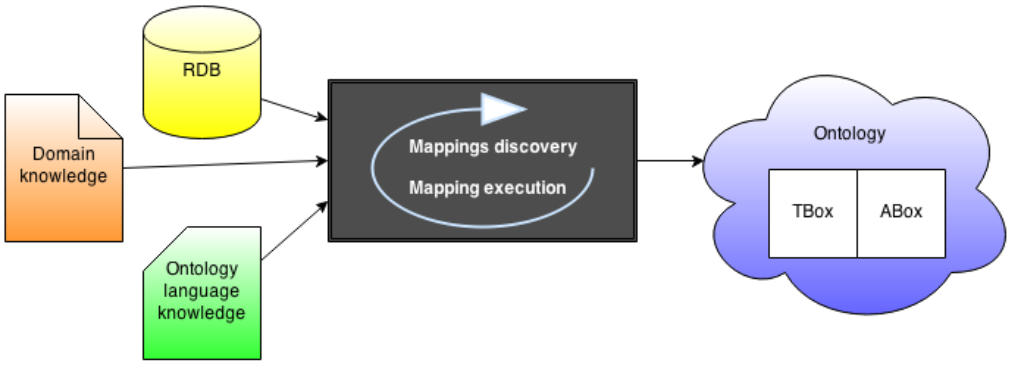

Fig. 14. The process

Moreover, the following steps need to be carried out:

1. Preparation of the relational database: refactoring, normalization, data cleaning (SKIPPED)
2. Analysis and development of a conceptual model or desired destination ontology (SKIPPED: THE WORK OF PREVIOUS ASSIGNMENT IS REUSED)
3. Direct mapping generation
4. Refinement and enrichment of the mappings
5. Implementation of the mapping procedure (manually or using a tool) to produce an ontology as a result

6.2 Inputs

The schema and the data of the sample relational database is shown, respectively, in Figure 15 and 16.

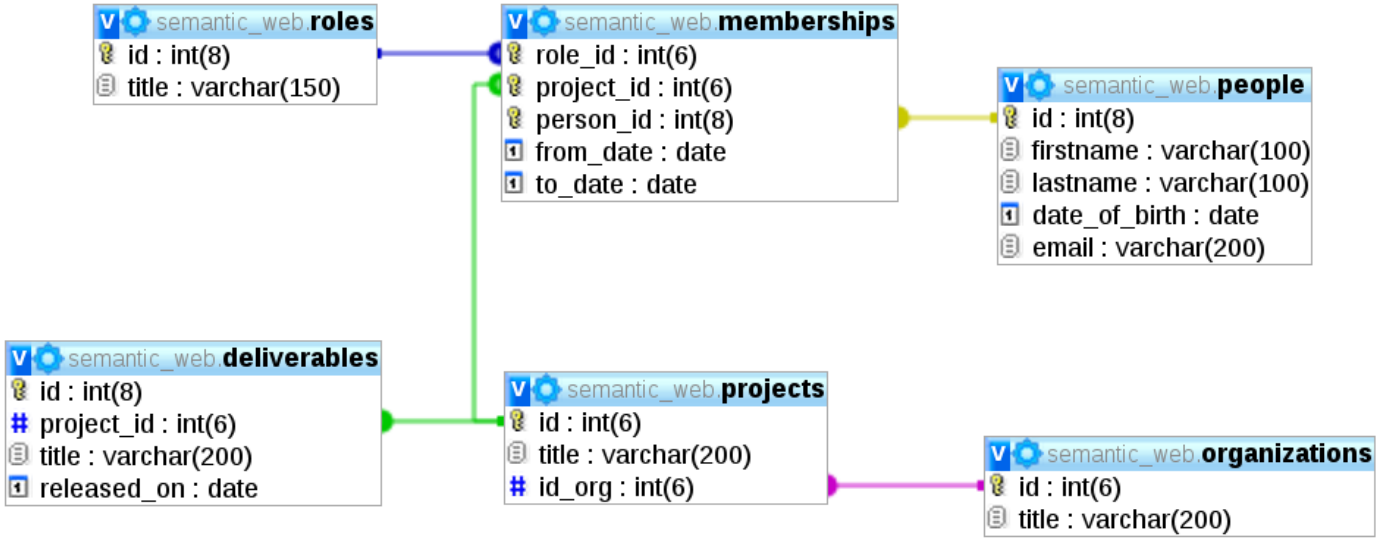

Fig. 15. The source schema for our examples

SQL query: SELECT * FROM `deliverables` LIMIT 0, 30 ;
Rows: 3

id	project_id	title	released_on
1	3 [->]	This Database	2014-06-15
2	3 [->]	This Ontology	2014-06-15
3	3 [->]	This Report	2014-06-15

SQL query: SELECT * FROM `organizations` LIMIT 0, 30 ;
Rows: 2

id	title
1	Unibo
2	MyOrganization

SQL query: SELECT * FROM `memberships` LIMIT 0, 30 ;
Rows: 2

role_id	project_id	person_id	from_date	to_date
1 [->]	3 [->]	2 [->]	2014-06-01	2014-06-15
2 [->]	4 [->]	3 [->]	*NULL*	*NULL*

SQL query: SELECT * FROM `projects` LIMIT 0, 30 ;
Rows: 2

id	title	id_org
3	My Assignment	2 [->]
4	Teaching at Unibo	1 [->]

SQL query: SELECT * FROM `roles` LIMIT 0, 30 ;
Rows: 4

id	title
1	Project manager
2	Customer
3	Stakeholder
4	Factotum

SQL query: SELECT * FROM `people` LIMIT 0, 30 ;
Rows: 2

id	firstname	lastname	date_of_birth	email
2	Roberto	Casadei	*NULL*	roberto.casadei12@studio.unibo.it
3	Antonella	Carbonaro	*NULL*	antonella.carbonaro@unibo.it

Fig. 16. The source data for our examples

Note that there are two kinds of instance data. One kind is relative to ontology individuals. The other kind is about project-specific data which does not belong to the ontology but to an actual project management effort.

6.3 Direct mapping into RDFS/OWL

For the purpose, I use the D2RQ tool[1].

```
d2rq/$ ./generate-mapping --w3c -v -u <user> -p <pass>
       jdbc:mysql://localhost/semantic_web
```

which produces the following output:

[1] http://d2rq.org/

```
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix :        <vocab/> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:organizations_id
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :organizations ;
      rdfs:label "organizations_id" ;
      rdfs:range xsd:integer .

:deliverables_id
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :deliverables ;
      rdfs:label "deliverables_id" ;
      rdfs:range xsd:integer .

:people_firstname
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :people ;
      rdfs:label "people_firstname" ;
      rdfs:range xsd:string .

:deliverables
      a       owl:Class , rdfs:Class ;
      rdfs:label "deliverables" .

:projects_id
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :projects ;
      rdfs:label "projects_id" ;
      rdfs:range xsd:integer .

:projects_title
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :projects ;
      rdfs:label "projects_title" ;
      rdfs:range xsd:string .

:memberships_role_id
      a       rdf:Property , owl:DatatypeProperty , owl:ObjectProperty ;
      rdfs:domain :memberships ;
      rdfs:label "memberships_role_id" ;
      rdfs:range xsd:integer , :roles .

:memberships
      a       owl:Class , rdfs:Class ;
      rdfs:label "memberships" .

:memberships_project_id
      a       rdf:Property , owl:DatatypeProperty , owl:ObjectProperty ;
      rdfs:domain :memberships ;
      rdfs:label "memberships_project_id" ;
      rdfs:range :projects , xsd:integer .

:people_lastname
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :people ;
      rdfs:label "people_lastname" ;
      rdfs:range xsd:string .

:people_email
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :people ;
      rdfs:label "people_email" ;
      rdfs:range xsd:string .

:people
      a       owl:Class , rdfs:Class ;
      rdfs:label "people" .

:people_id
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :people ;
      rdfs:label "people_id" ;
      rdfs:range xsd:integer .

:roles_id
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :roles ;
      rdfs:label "roles_id" ;
      rdfs:range xsd:integer .

:roles
      a       owl:Class , rdfs:Class ;
      rdfs:label "roles" .

:roles_title
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :roles ;
      rdfs:label "roles_title" ;
      rdfs:range xsd:string .

:organizations_title
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :organizations ;
      rdfs:label "organizations_title" ;
      rdfs:range xsd:string .

:organizations
      a       owl:Class , rdfs:Class ;
```

```
      rdfs:label "organizations" .

:memberships_to_date
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :memberships ;
      rdfs:label "memberships_to_date" ;
      rdfs:range xsd:date .

:deliverables_released_on
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :deliverables ;
      rdfs:label "deliverables_released_on" ;
      rdfs:range xsd:date .

:memberships_person_id
      a       rdf:Property , owl:DatatypeProperty , owl:ObjectProperty ;
      rdfs:domain :memberships ;
      rdfs:label "memberships_person_id" ;
      rdfs:range :people , xsd:integer .

:projects_id_org
      a       rdf:Property , owl:DatatypeProperty , owl:ObjectProperty ;
      rdfs:domain :projects ;
      rdfs:label "projects_id_org" ;
      rdfs:range :organizations , xsd:integer .

:projects
      a       owl:Class , rdfs:Class ;
      rdfs:label "projects" .

:memberships_from_date
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :memberships ;
      rdfs:label "memberships_from_date" ;
      rdfs:range xsd:date .

:people_date_of_birth
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :people ;
            rdfs:label "people_date_of_birth" ;
      rdfs:range xsd:date .

:deliverables_title
      a       rdf:Property , owl:DatatypeProperty ;
      rdfs:domain :deliverables ;
      rdfs:label "deliverables_title" ;
      rdfs:range xsd:string .

:deliverables_project_id
      a       rdf:Property , owl:DatatypeProperty , owl:ObjectProperty ;
      rdfs:domain :deliverables ;
      rdfs:label "deliverables_project_id" ;
      rdfs:range :projects , xsd:integer .
```

6.4 Developing upon the direct mappings

The direct mappings are just a starting point. In fact, they need to be refined and extended because they are flawed in many aspects:

- We are not interested in exposing surrogate keys (such as `deliverables_id`, `people_id`, ...) as properties
- The `title` properties may be seen just as `rdfs:labels` for individuals and thus can be pruned
- We would like to reuse existing ontologies when possible: so, for example, the `organizations` table/class should be mapped to `org:Organization` and `people_firstname` should be mapped to `foaf:firstName` (e.g., by making the `people` table/class a subclass of `foaf:person`)
- The names are quite ugly and we may want to make them in singular form (e.g., from `people` table to `person` concept)

The aforementioned changes are manually applied and the resulting OWL ontology is reported below:

```
@prefix :        <baseuri> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix org: <http://www.w3.org/ns/org#> .

:PMConcept
      a       owl:Class, skos:Concept .

:Person
      a       owl:Class, :PMConcept ;
      rdfs:subClassOf foaf:Person .
#:firstname
```

```
#      a       owl:DatatypeProperty ;
#      rdfs:subPropertyOf foaf:firstName .

:Role
      a       owl:Class, :PMConcept  .

:Organization
      a       owl:Class, :PMConcept  ;
      rdfs:subClassOf org:Organization .

:Membership
      a       owl:Class, :PMConcept ;
      rdfs:subClassOf org:Membership .
:within
      a       owl:ObjectProperty ;
      rdfs:domain :Membership ;
      rdfs:range :Project.

:Deliverable
      a       owl:Class, :PMConcept .
:released_on
      a       owl:DatatypeProperty ;
      rdfs:domain :Deliverable ;
      rdfs:range xsd:date .
:outputOf
      a       owl:ObjectProperty ;
      rdfs:domain :Deliverable ;
      rdfs:range :Project .

:Project
      a       owl:Class, :PMConcept .
:startedBy
      a       owl:ObjectProperty ;
      rdfs:domain :Project ;
      rdfs:range :Organization .
```

References

1. J. Ravi, Z. Yu, and W. Shi, "A survey on dynamic web content generation and delivery techniques," *J. Netw. Comput. Appl.*, vol. 32, pp. 943–960, Sept. 2009.
2. D.-E. Spanos, P. Stavrou, and N. Mitrou, "Bringing relational databases into the semantic web: A survey," *Semant. web*, vol. 3, pp. 169–209, Apr. 2012.
3. J. Sequeda, "Relational database to rdf - rdb2rdf." http://vimeo.com/66718408.
4. A. Soren, L. Feigenbaum, D. Miranker, A. Fogarolli, and J. Sequeda, "Use cases and requirements for mapping relational databases to rdf." http://www.w3.org/TR/rdb2rdf-ucr/.
5. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau Jr, S. Auer, J. Sequeda, and A. Ezzat, "A survey of current approaches for mapping of relational databases to rdf." http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf.
6. M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda, "A direct mapping of relational data to rdf." http://www.w3.org/TR/rdb-direct-mapping/.
7. S. Das, S. Sundara, and R. Cyganiak, "R2rml: Rdb to rdf mapping language." http://www.w3.org/TR/r2rml/.
8. J. Bakkas, M. Bahaj, and A. Marzouk, "Article: Direct migration method of rdb to ontology while keeping semantics," *International Journal of Computer Applications*, vol. 65, pp. 6–10, March 2013. Published by Foundation of Computer Science, New York, USA.
9. N. Gherabi and K. Addakiri, "Mapping relational database into owl structure with data semantic preservation," *(IJCSIS) International Journal of Computer Science and Information Security Vol. 10, No. 1.*
10. A. Marzouk, "Article: Mapping process of relational schema to owl ontologies using xslt," *International Journal of Computer Applications*, vol. 71, pp. 48–54, June 2013. Published by Foundation of Computer Science, New York, USA.
11. J. Sequeda, "Relational database and the semantic web." http://semanticweb.com/relational-database-and-the-semantic-web_b16083.