

# A Framework for Transforming Archaeological Databases to Ontological Datasets

**Yi Hong, Monika Solanki,**  
Department of Computer Science  
University of Leicester  
{yh37, ms491}@le.ac.uk

**Lin Foxhall, Alessandro Quercia**  
School of Archaeology and Ancient History  
University of Leicester  
{lf4, aq15}@le.ac.uk

## 1. Introduction

The benefits of disseminating archaeological data via the web have been highlighted as early as 2001 [11]. The potential impact of Semantic Web [5] on archaeology has also been well recognised [9]. Traditionally, archaeological experts are used to recording data in relational databases. This is usually an SQL repository or the so called "Deep web"[4]. These repositories are only accessible via web-based end points provided by the holders of the data sources. The majority of the rich and contextually relevant data, which would be of benefit a larger community, lies buried underneath layers of application interfaces. Indeed, relational data silos provide scalable storage and efficient query execution mechanism but they are inaccessible to the most popular and powerful web-based search engines.

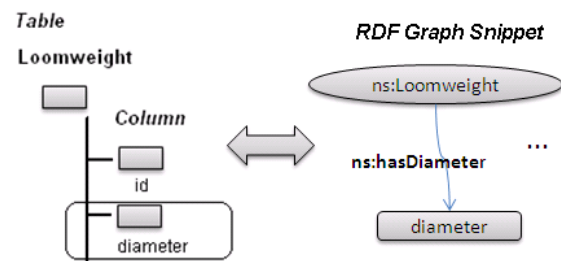
To increase the uptake and usage of archaeological data, these resources need to be openly available and accessible both to humans and machines. The Semantic Web provides the infrastructure that is necessary for data to be smartly marked up and made available as ontologies. Archaeological data exposed as ontologies immediately open up the domain to the extensive suite of semantic web aware applications such as data browsers and search engines. Vast amounts of fragmented archaeological data could thus be systematically structured and made available to a wider community.

One such project which is expected to generate large volumes of data stored in RDBs (relational databases) is the Leverhulme funded *Tracing Networks* research programme [1]. The project aims to trace the links between people involved in the production, consumption and distribution of material artifacts across and beyond the Mediterranean region. The project encompasses six closely-linked subprojects dealing with artifacts such as loomweights, lefkandi pottery, crafts at Tiryns, coinage, punic ceramics and human representations in art.

In this paper we propose a framework for the transformation of archaeological data sources in the project to RDF [16] data models. Relational data sources are first transformed into data objects. These objects are then mapped into ontology instances using an ECA-based [15] scripting language. The paper is structured as follows: Section 2 discusses the problem with existing mapping approaches. Section 3 presents our proposed transformation framework. Section 4 illustrates our prototype implementation. Section 5 discusses related work and Section 6 presents conclusions and future work.

## 2. The Problem with Mapping

Typically, when converting data from a RDB to ontologies, the scripting languages conventionally used provide simplistic mapping rules. Existing frameworks [7, 8] map table names to RDF/OWL classes and columns to properties in the ontology schema. However they provide limited support in terms of what can be mapped. Generally, the mapping specified is one-to-one, i.e., one column mapped to one concept, as illustrated in *Figure 1*.



**Figure 1:** One-to-one mapping example

In most scenarios, the association between columns and properties is far more complex than a simple one-to-one correspondence. This may happen if the domain specific schemas to be used for mapping have been extended from standard vocabularies or those used elsewhere. For example, in this paper, rather than simply generating arbitrary RDF/OWL instances from relational data sources, we would like our ontological instances to conform to a domain specific

schema, e.g. CIDOC-CRM [2] for the archaeological domain. It could also be the case that several ontology schemas are used and the data needs to be suitably mapped to more than one property.

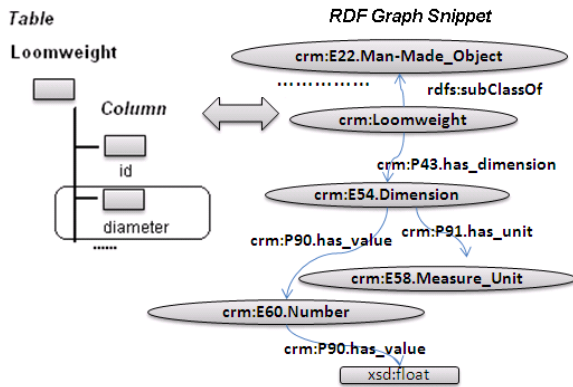


Figure 2: Example of relationship between table and the CIDOC CRM ontology

As an example, we consider data from a subproject of the Tracing Networks research programme. The project investigates *loomweights* across Mediterranean societies as an archaeological artifact. A loomweight is characterised by attributes such as number of holes, stamps (impressions) thickness and diameter. Figure 3 illustrates a loomweight.



Figure 3: A two-hole loomweight with imprinted meander and wave design.

The ontology schema used for the transformation has been extended from CIDOC-CRM. This is illustrated in Figure 2. In this figure, it can be seen that the column *diameter* of the RDB table cannot be mapped in a straightforward manner, i.e., as a data-type property, in the schema. In order to specify a relationship between *diameter* and the concept *Loomweight* we have to create several intermediate instances of concepts defined by CIDOC-CRM. These instances have to be contextually related to each other in order to ensure that Loomweights are assigned correct diameter values.

### 3. Transformation Framework

To address the problems of mapping data sources, we propose a new framework for transforming archaeological RDBs to ontological datasets such as CIDOC-CRM, an ISO standard for the integration of cultural and heritage information. The transformation

workflow, as depicted in Figure 1, consists of three main steps:

- (1) ORM Reverse Engineering.
- (2) ECA Rule-based Transformation.
- (3) Ontology Instance Generation.

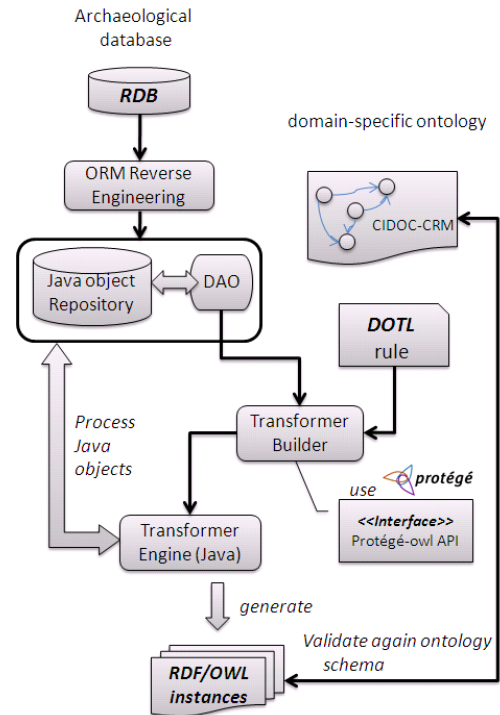


Figure 4: Transformation Framework.

#### 3.1 ORM Reverse Engineering

The ORM (Object-Relational Mapping) technique is commonly used for storing persistent object-oriented data within heterogeneous RDBs. On the other hand, the ORM Reverse Engineering[13] technique is able to extract the existing tables, columns, relationships (inc. primary key, foreign key, join etc) and index from a relational database to object-oriented data structures. It has the capability of representing an RDB table as data structures or “classes”. Therefore, records in the table can be instantiated as data objects, which can be easily manipulated and processed using Object-Oriented Programming (OOP).

Since we have a huge amount of data distributed across several different legacy RDBs, e.g., MySQL and MS Access, amongst others, the transformation framework must be designed to be able to flexibly connect to various databases to retrieve the data. One of the most popular and compatible data persistence solutions is the open source Hibernate ORM [12] framework. In our proposed approach, the Hibernate ORM Reverse Engineering tool is used to convert database records into Java objects. Technical details

and database compatibility issues in this regards are further discussed in section 4.

### 3.2 ECA Transformation Rules

After converting database records into object-oriented data, the next and the most important step is to define transformation rules for these objects. The conventional mapping mechanism defines mapping rules in their own language such as D2RQ[7] and Virtuoso[8]. This approach basically maps tables to RDF/OWL classes, columns to properties, which works perfectly fine with straight forward one- to-one mapping we discussed in *Figure 1*.

This approach lacks the capability of expressing complex non-linear relationship as shown in *Figure 2*. However, this type of mapping is very common when transforming databases to sophisticated target ontologies such as CIDOC-CRM.

In our case, we developed an ECA-based [15] (Event-Condition-Action) textual transformation language *DOTL* — *Database Ontology Transformation Language*. Unlike the existing mapping mechanisms, which merely specify one-to-one mappings at schema level, the fundamental construct of a *DOTL* transformation rule is of form:

*On Event if Condition Do Action*

A basic *DOTL* rule consists of three parts: (1) The *event* part specifies the triggers of the transformation rule, usually the occurrence of an object of a specific class; (2) The *condition* part is a logical expression, which checks the pre-condition of the action to be carried out, the default conditions being “if undefined”; (3) The *action* part usually consists of a series of creation of new ontology instances, properties and other corresponding operations.

```
foreach item:Loomweight do{
  createOWLInstance loomweight
    type "crm:loomweight"
    URI "crm:loomweight_" + item.id

  onFeature item.diameter {
    createRDFInstance dimension
      type "crm:E54.Dimension"
      URI "crm:E54.Dimension_" + item.id

    createRDFInstance number
      type "crm:E54.Number"

    createRDFInstance cm
      type "crm:E58.Measurement_unit"
      URI "crm:E58.Measurement_unit_cm"+item.id

    createPrimitive integer
      type "Integer"
      value item.diameter

    createOWLProperty ("tn:hasMaxHoleDiameter",loomweight,dimension)
    createRDFProperty ("crm:P90.has_value",dimension,number)
    createRDFProperty ("crm:has_FloatNumber",number,integer)
    createRDFProperty ("crm:P91.has_unit",dimension,cm)
  }
}
```

**Listing 1:** *DOTL transformation rules*

The *DOTL* code snippet in *Listing 1* outlines the mapping rules for the *Loomweight* table when the transformation procedures illustrated in *Figure 3* are

applied to the RDB sources: for each *Loomweight* element in the object collection, create a corresponding RDF *Loomweight* individual; if the value of attribute “*diameter*” is not null, then perform a sequence of operations as listed in *Listing 1*, such as creating a new instance of intermediate class *Dimension*, specifying the URI pattern, establishing a link between *Dimension* and *Loomweight* instance, assigning value to appropriate data type property.

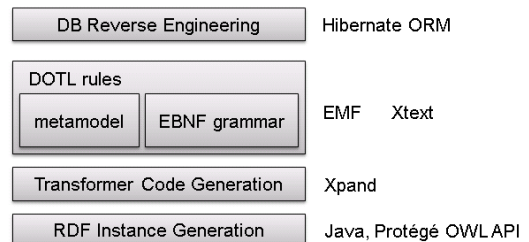
### 3.3 Instance Generation

Once the transformation rules are specified, the last step is to generate ontology instances accordingly. Instead of creating a virtual in-memory model on the fly, the framework exports all data to the RDF store or as persistent RDF files. *DOTL* do not use a compiler to get the executable binary code. Instead, *Transformer builder* component is used to build Java source code based on the pre-defined *DOTL* rules we specified in the previous steps. Finally the *Transformer Engine* component compiles and executes the Java code to generate RDF/OWL instances. The generation of ontology instances from relational database is now complete.

## 4. Implementation

A prototype implementation of the framework has been developed in Java. The prototype uses the open source Hibernate Reverse Engineering framework for object/relational mapping, which provides connectivity support for Oracle, DB2, SQL Server, MySQL and most mainstream relational databases. In this way, we are able to retrieve data from various existing archaeological databases as well as integrate data sources in distributed RDBs.

A textual *DOTL* Editor plugin for Eclipse has also been implemented, which supports syntax coloring, code completion, syntax checking/error markers, and navigation. Formalised EBNF grammar of *DOTL* is defined in Xtext [14] syntax while the metamodel of the language is described using the EMF [14] (Eclipse Modeling Framework). The *DOTL* Editor also contains an integrated Java code generator implemented in Xpand [14] for building executable transformation program. Generated Java code invokes Protégé-OWL API to perform the generation of RDF/OWL instances.



**Figure 5:** *Layers and Implementation Techniques*

Figure 5 illustrates the layers of the transformation framework and relevant techniques used during the prototype implementation.

## 5. Related work

A survey of approaches for mapping RDB to RDF is presented in [10]. D2RQ [7] and Virtuoso [8] provide a declarative language to describe mappings between relational database schemata and OWL/RDFS ontologies. The language is limited to specifying basic triple pattern match. An approach closely related to our work is R2O [3]. The authors specify an XML-based language for the transformation. In related archaeology projects [17], the creation of initial mappings between database columns and RDF entities was a manual exercise undertaken with the benefit of domain knowledge from English Heritage. In [6] bespoke tools are provided that guide the data curators through the mapping process, using basic natural language processing. In our approach we can specify complex mapping patterns for ontology transformation. RDB schemas can be quite complicated in terms of integrity constraints. Our framework provides support for dealing with many of these constraints, since we build our work on the Hibernate framework.

## 6. Conclusion and Future work

In this paper we proposed a transformation framework for migrating large volumes of archaeological data stored in RDBs to ontology based data sets on the Semantic Web. We proposed the ECA-based scripting language DOTL, which allows the specification of complex transformation rules from data objects to ontologies. We discussed a motivating example based on the CIDOC-CRM ontology schema as a case study. Finally we presented a prototype implementation that illustrates our methodology.

We are currently refining the grammar and semantics to enhance the expressiveness of DOTL to improve the usability of the system. Another ongoing development is to implement a user-friendly graphical modeling environment for the language in GMF (Graphical Modeling Framework) to allow easy creation and editing of transformation rules.

**Acknowledgement.** This work was partially supported by the Leverhulme Trust Programme Award “Tracing Networks”.

## References

[1] Tracing Networks Research Programme: Craft Traditions in the Ancient Mediterranean and Beyond, <http://www.tracingnetworks.ac.uk>

[2] The CIDOC Conceptual Reference Model

<http://cidoc.ics.forth.gr>

[3] Jesús Barras, Óscar Corcho and Asunción Gómez-pérez. 2004, R2O, An Extensible and Semantically based Database-to-Ontology Mapping Language

[4] Michael K Bergman. The Deep Web: Surfacing Hidden Value.

[http://www.brightplanet.com/images/uploads/DeepWebWhitePaper\\_20091015.pdf](http://www.brightplanet.com/images/uploads/DeepWebWhitePaper_20091015.pdf)

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web, 2001, *Scientific American Magazine*

[6] Leif Isaksen and Kirk Martinez and Nicholas Gibbins and Graeme Earl and Simon Keay. 2009. Linking Archaeological Data, *Computer Applications and Quantitative Methods in Archaeology conference*

[7] Christian Bizer and Andy Seaborne. 2004, D2rq - treating non-rdf databases as virtual RDF graphs. In *ISWC2004 (posters)*.

[8] Orri Erling and Ivan Mikhailov. 2007, RDF support in the Virtuoso DBMS. In *Conference on Social Semantic Web*.

[9] Julian D. Richards. 2006, Archaeology, e-publication and the semantic web. *Antiquity*, (310).

[10] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda and Ahmed Ezzat. 2009, *A Survey of Current Approaches for Mapping of Relational Databases to RDF*.

[11] J. David Scholen. Archaeological data models and web publication using xml. 2001, *Computers and the Humanities*, 35(3).

[12] Red Hat Middleware, 2009, “Relational Persistence for Java and .NET”, [www.hibernate.org](http://www.hibernate.org)

[13] Astrova, I., 2004. Reverse Engineering of Relational Databases to Ontologies, *The Semantic Web: Research and Applications*, pp: 327-341, LNCS

[14] Gronback R.C, 2008. Eclipse Modeling Project – A Domain-Specific Language (DSL) Toolkit, pp 277-313, 605-649, Addison-Wesley Pearson Education

[15] James Bailey, Alexandra Poulouvassilis and Peter T. Wood, 2002. An Event-Condition-Action language for XML, pp: 486 - 495, In Proceedings of *The 11th international conference on World Wide Web*

[16] Resource Description Framework (RDF), <http://www.w3.org/RDF>

[17] Binding, Ceri, May, Keith and Tudhope, Douglas, 2008, Semantic Interoperability in Archaeological Datasets: Data Mapping and Extraction Via the CIDOC CRM, ECDL '08: *12th European conference on Research and Advanced Technology for Digital Libraries*