Exploiting Aggregated Open Data from Smart Cities for the Future Internet Society

D3.4: SMART-FI Semantic Modelling Techniques Report v2

Authors	Serdar Yumlu, Mehmet Kurt, Cihat Yigit, Ehsan Valizadeh
Institution lead	SAMPAS
Version	V1.0
Reviewers	Caner Tosunoglu (SAMPAS)
Work package	WP3
Task	T3.2
Due date	30/04/2018
Submission date	07/09/2018
Distribution level (CO, PU):	PUBLIC
Abstract	These reports will analyze the set of techniques for semantic modeling.
Keywords	Semantic Modeling, Relational Database, Smart Cities Ontologies, Database-to-Ontology Mapping
License information	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Description of change	*Modified by*
v0.1	01/02/18	First draft version and TOC	Serdar Yumlu (SAMPAS)
v0.2	15/02/18	contributions	Cihat Yigit (SAMPAS)
v0.3	03/03/18	contributions	Mehmet Kurt (SAMPAS)
V0.4	06/03/18	contributions	Ehsan Valizadeh (SAMPAS)
V0.5	25/03/18	contributions	Caner Tosunoglu (SAMPAS)
V0.6	18/04/18	Sent to reviewer	Caner Tosunoglu (SAMPAS)
V0.7	10/05/18	Check the status and contributions	Ehsan Valizadeh (SAMPAS)
V0.8	25/06/18	Check the status and contributions	Caner Tosunoglu (SAMPAS)
V0.9	15/07/18	Check the status and contributions	Caner Tosunoglu (SAMPAS)
V1.0	07/09/18	Overall revision of document and prepare final version to release	Ehsan Valizadeh (SAMPAS)

Table of Contents

Table of Contents 3
Table of Figures 4
Table of Tables 4
Terms and abbreviations 5
Executive Summary 6
1 Introduction 7
1.1 About this deliverable 7
1.2 Document structure 7
2 Reference Framework for Survey of conversion of a RDB into ontology 8
2.1 Components of Survey Framework 8
A. Automatic Mapping Generation 8
B. Domain Semantics-driven Mapping Generation 8
C. Mapping Representation and Accessibility of Mappings 9
D. Mapping Implementation 9
E. Query Implementation 9
F. Application Domain 9
G. Data Integration 9
3 Survey of RDB to RDF Mapping Approaches and Conversion Tools 10
3.1 Proof of Concept Projects 10
3.2 Domain-specific projects 11
3.3 Tools/Applications 12
3.3.1 DB2OWL 12
3.3.2 R2O 12
3.3.3 D2RQ 12
3.3.4 Data Semantic Preservation 12
3.3.5 DartGrid Semantic 12
3.3.6 Asio Semantic Bridge for Relational Databases (SBDR) and Automapper 12
3.3.7 XTR-RTO 13
3.3.8 RTAXON 14
3.3.9 Ontology Generator (RDB2On) 14
3.3.10 RDBToOnto 14
3.3.11 OntoEdit 15
3.3.12 WebODE 15
3.3.13 Protégé-2000 16
3.3.14 OE: Ontology editor in Hozo 17
3.3.15 RDB2RDF 18
A. Transformation Approaches 18

B. List of Available Tools 19
4 Conclusion 20
5 References 20

Table of Figures

FIGURE 1 REFERENCE ARCHITECTURE FOR RDB2RDF SURVEY 10
FIGURE 2 BOOK TABLE IN RELATIONAL MODEL 13
FIGURE 3 RELATIONS IN THE ONTOLOGY 13
FIGURE 4 OWL DESCRIPTION OF BOOK TABLE 14

Table of Tables

No table of figures entries found.

Terms and abbreviations

RDF	Resource Description Framework
SPARQL	RDF query language
ETL	Extract, Transform and Load
LOD	Linked Open Data cloud
URI	Uniform Resource Identifier
EAI	Enterprise Application Integration
GE	Generic Enabler
OWL	Web Ontology Language
WP	Work Package

Executive Summary

First version of this deliverable focused on available semantic modeling techniques like ontologies and details of an ontology, an introduction to ontology engineering and then assessing several sample smart city ontologies.

As second version of deliverable, the objective of this document is to examine and present a comparative analysis of proposed tools and algorithms that enabled the automatic conversion of a relational database into ontology. While conceptual mapping rules/principles of relational databases and ontology structures are being proposed, several software modules or plug-ins are being examined to enable the automatic and manual conversion of relational databases into ontologies.

1 Introduction

This document surveys current techniques, tools and applications for mapping between Relational Databases and the Resource Description Framework (RDF).

While Ontology construction from a relational database used to be a manual and tedious process which relied solely on ontology editors and human experts, many prototype stage tools are available to convert a relational database into ontology automatically. Examples of such tools and algorithms include: DB2OWL, R2O, D2RQ, Data Semantic Preservation, DartGrid Semantic, Semantic Bridge, Automapper, XTR-RTO, RTAXON, Leaning Ontology from Relational Databases, Ontology Generator (RDB2On), and RDBToOnto amongst others. The World Wide Web Consortium (W3C) through their RDB2RDF Working Group are also developing a direct mapping standard that focuses on translating relational database into RDF (Resource Description Framework) ontology.

1.1 About this deliverable

A critical requirement for the evolution of the current Web of documents into a Web of Data (and ultimately a Semantic Web) is the inclusion of the vast quantities of data stored in Relational Databases (RDB). The mapping of these vast quantities of data from RDB to the Resource Description Framework (RDF) has been the focus of a large body of research work in diverse domains and has led to the implementation of generic mapping tools as well as domain-specific applications. Furthermore, the role of RDF as an integration platform for data from multiple sources, primarily in form of RDB, is one of the main motivations driving research efforts in mapping RDB to RDF. This survey documents current tools and approaches in mapping RDB to RDF and effectively categorizes and compares the different approaches using a well-defined reference framework.

This study reviews and applies abovementioned tools to automatically construct ontologies from a relational database. The resulting ontologies are further analyzed to match their structures against the database-to-ontology mapping principles.

1.2 Document structure

Section 1 introduces the document; Section 2 contains "Reference Framework for Survey of conversion of a RDB into ontology"; Section 3 lists "RDB to RDF Mapping Approaches and Conversion Tools"; Section 4 is conclusion of deliverable; Finally, section 5 includes References.

2 Reference Framework for Survey of conversion of a RDB into ontology

A reference framework will enable the effective categorization and comparison of different approaches used in converting RDB data to RDF. We introduce a reference framework for this survey consisting of several components such as the approach to generate the RDB to RDF mapping, the representation and the achieved integration of data.

2.1 Components of Survey Framework

In this section we describe each of the components of the reference framework in detail. We can classify the methods used to generate the mappings between RDB and RDF into two categories:

A. Automatic Mapping Generation

As a a set of mappings between RDB and RDF namely:

1. A RDB record is a RDF node
2. The column name of a RDB table is a RDF predicate
3. A RDB table cell is a value

Many systems leverage these mappings to automatically generate mappings between RDB and RDF with the RDB table as a RDF class node and the RDB column names as RDF predicates. An example of this approach is the Virtuoso RDF View that uses the unique identifier of a record (primary key) as the RDF object, the column of a table as RDF predicate and the column value as the RDF subject. Other examples of similar tools are D2RQ (D2RQ also allows users to define customized mappings) and SquirrelRDF.

Even though these automatically generated mappings often do not capture complex domain semantics that are required by many applcations, these mappings serve as a useful starting point to create more customized, domain-specific mappings. This approach also enables Semantic Web applications to query RDB sources where the application semantics are defined in terms of the RDB schema. This approach is also called "Local ontology mapping".

Use of Existing Ontology Schema: A variation of the automatic generation of mappings between RDB and RDF is the use of an existing ontology to create simple mapings. These simple mappings are checked for consistency and subsequently more contextual mappings are constructed. Though this approach automatically generates the RDB to RDF mappings, an existing ontology is used to enhance the quality of the mappings.

B. Domain Semantics-driven Mapping Generation

The second approach generates mappings from RDB to RDF by incorporating domain semantics that is often implicit or not capture at all in RDB schema. The explicit modeling of domain semantics, often modeled as a domain ontology, in the RDF repository enables software applications to take advantage of this "information gain" and execute queries that link together entities such as "gene --> expressed_in --> brain". Additionally, a mapping generated by using domain semantics also reduces the creation of triples for redundant or irrelevant knowledge.

The domain ontology may be pre-existing and sourced from public resources such as the National Center for Biomedical Ontologies (NCBO) at http://bioportal.bioontology.org/ or may be bootstrapped from local ontologies created by automatic mapping tools (as described in previous section). Green et al. discuss an approach of mapping spatial data to RDF using a hydrology ontology as the reference knowledge model and Sahoo et al. discuss an approach to generate mappings using the Entrez Knowledge Model (EKoM) to map gene data to RDF.

This approach is also called "Domain ontology mapping" and the process is the same as an ontology population technique where the transformed data are instances of the concepts defind in the ontology schema. Many mapping tools such as D2RQ allow the users to create customized mapping rules in addition to the automatically generated rules.

C. Mapping Representation and Accessibility of Mappings

The mappings between RDB and RDF may be represented as XPath rules in a XSLT stylesheet, in a XML-based declarative language such as R2O or as "quad patterns" defined in Virtuoso meta-schema language. The mappings, especially if they are created by domain experts or reference a domain ontology, may have wider applicability. Hence, to encourage reuse the mappings should be easily accessible by the wider community. Consequently, we review the representation and accessibility of the mappings between RDB and RDF.

D. Mapping Implementation

The actual mapping of RDB to RDF may be either a static Extract Transform Load (ETL) implementation or a query-driven dynamic implementation. The ETL implementation such as the application described by Byrne, also called "RDF dump", use a batch process to create the RDF repository from RDB using mapping rules. The dynamic approach, an example application is described by Green et al., implements the mapping dynamically in response to a query.

Similar to the data warehouse implementations the ETL approach may not reflect the most current data, but it allows additional processing or analysis over the data including execution of inference rules (well-defined RDF entailments or user-defined rules) without compromising query performance. On the other hand, the dynamic approach has an advantage that the query is executed against the most current version of data, which assumes significance if the data is frequently updated. However, a dynamic mapping implementation may incur query performance penalties if entailment rules are applied to the RDF repository to infer new information.

E. Query Implementation

Queries in systems mapping RDB to RDF may either be in SPARQL, which is executed against the RDF repository, or the SPARQL query may be transformed into one or more SQL queries that are executed against the RDB. Cyganiak has discussed the transformation of SPARQL to relational algebra and further into SQL. This paper describes operators such as "selection" and "inner join" implemented over RDF and correlates "RDF relational algebra" to SQL.

F. Application Domain

As discussed earlier in "Creation of Mappings" section, an important aspect of mapping RDB to RDF is the incorporation of domain semantics in the resulting RDF. Hence, we explicitly list the application domain of the work reviewed in this survey where possible (mapping tools are not domain-specific).

G. Data Integration

The RDF representation model through use of URI and explicitly modeled relationships between entities makes it easier to effecively integrate data from disparate, heterogeneous data sources. It is important to note that RDF does not automatically reconcile the multiple types of heterogeneity, such as structural, syntactic and semantic heterogeneities, that are described in the data/information integration research field. But, the use of domain ontologies along user-defined inference rules in reconciling heterogeneity between multiple RDB sources is an effective approach for creating a single or a set of "comaptible" RDF. Hence, this metric reviews the different mapping approaches with respect to data integration.

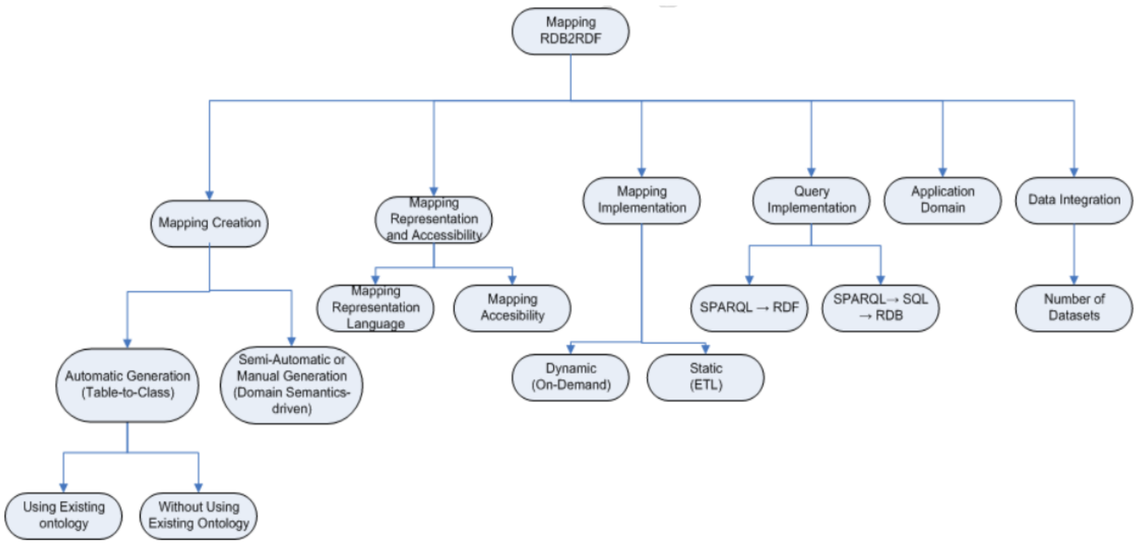


Figure 1 Reference Architecture for RDB2RDF Survey

3 Survey of RDB to RDF Mapping Approaches and Conversion Tools

In this section, we categorize the surveyed work into three broad classes namely:

1. **Proof of concept projects**: Projects reviewed in this section explore specific approaches to map RDB to RDF with either a prototype or proof of concept imlementation. The work may or may not have lead to the release of a tool/application to the community
2. **Domain-specific projects**: Many projects surveyed in this paper were driven by real-world application requirements and have used domain-semantics based customized mappings, generic mapping tools or a combination of both
3. **Tools/Applications**: The projects surveyed in this section include D2RQ, R2O, Virtuoso, Triplify, and Dartgrid tools that have been released to the community for mapping RDB to RDF

All the projects in the three categories are reviewed according to the reference architecture defined above.

3.1 Proof of Concept Projects

1. Hu et al. describe an approach that aims at the automatic creation of simple mappings between relational database schemas and ontologies. Based on the relational schema and an ontology, initial simple mappings are derived and then checked for consistency. Based on sample input (mappings and instances for both the relational schema and the ontology) more contextual mappings are constructed. Experimental results in a limited domain showed the feasibility of the approach.

2. Kashyap et al. describe their work involving a mediator based approach to represent mappings from ontological concepts to disparate data soures as part of a general framework for RDF based access to heterogeneous data sources. The heterogeneous data sources, illustrated using a life sciences domain scenario, include RDB, Web services and Excel sheets (using MS Office API). The SPARQL queries are automatically translated to the appropriate query language using the mappings represented by the mediatr classes.
3. Cullot et al. describe DB2OWL using the Table to Class and Column to Predicate approach but use specific relational database schema characteistics, that is, how tables relate to each other, to assert subclass and other object properties. The object properties represent many-to-many relationships and referential integrity. The mappings are stored in a R2O document.
4. Tirmizi et al. follow Li et al. and other similar work. It is the first work to present formal rules in First Order Logic to translate Table to Class and Column to Predicate. A notion of completeness for a transformation system is also presented based on all the possible foreign key and primary key combinations.
5. The Semi-automatic Ontology Acquisition Method (SOAM) work by Li et al. uses the Table to Class and Column to Predicate approach to create an initial ontology schema which is then refined by referrig to a dictionary or thesauri (for example, WordNet). Constraints in the relational model are mapped to constraints in the ontology schema. For example, "NOT NULL" and "UNIQUE" are mapped to cardinality constraints on relevant properties. If a given set of relations are mapped to an ontology concept, the corresponding tuples of the relations are transformed as instances of the ontology concept.

3.2 Domain-specific projects

1. Sahoo et al. describe work in the life sciences domain that incorporates domain semantics (from multiple, integrated ontologies) to create the mappings (represented as XPath rules in XSLT stylesheet) from RDB to RDF. A RDF dump is created using a batch approach and stored in Oracle 11g. SPARQL query language is used to query the RDF repository.
2. Byrne describes an application in the cultural heritage domain to convert the National Monument Record of Scotland data stored in reational database to RDF. The Simple Knowledge Organization System (SKOS) framework is used to transform the Royal Commission on the Ancient and Historical Monuments of Scotland (RCAHMS) thesauri to the Semantic Web. The entire RCAHMS database with 1.5 million entities is converted to 21 million RDF triples and implemented on both Jena and AlegroGraph systems.
3. Green et al. describe integration of spatial data in RDB for predictive modeling of diffuse water pollution using OWL-DL ontologies at multi-levels. The ontologies ("Data ontologies") at the first level are used to map each data source to concepts in the ontologies at the next level ("Domain ontologies"). The "data ontologies" are represented in the D2RQ mapping language. The "application ontology", at the third level, links the "domain ontologies" and also adds application-specific information. The D2RQ engine is modified to include spatial operators and is used to interface between the data sources and data ontologies The SPARQL query language is used to query the virtual RDF graphs generated from the data sources.

3.3 Tools/Applications

3.3.1 DB2OWL

This tool considers particular table cases and take them into account while the mapping process. We have implemented a prototype of this tool in Java and using Jena3. This prototype deals currently with Oracle and MySQL databases because they provide specific views about the database metadata. Extension of the presented tool are underway to deal with other DBMS that provide such views. In addition, DB2OWL will be developed further to map several databases to one ontology, and to map databases from other models such as object-relational model.

3.3.2 R2O

R2O is a XML based declarative language to express the mappings between RDB elements and an ontology. R2O mappings can be used to "detect inconsistencies and ambiguities" in mapping definitions. The ODEMapster engine uses a R2O document to either execute the transformation in response to a query or in a batch mode to crate a RDF dump.

3.3.3 D2RQ

D2RQ provides an integrated environment with multiple options to access relational data including "RDF dumps", Jena and Sesame API based access (API calls are rewritten to SQL), and SPARQL endpoints on D2RQ Server.

The mappings may be defined by the user thereby allowing the incorporation of domain semantics in the mapping process, though there are some limitations to this as described in the Ordnance Survey presentation. The mappings are expressed in a "declarative mapping language". The performance varies depending on the access method and is reported to perform reasonably well for basic triple patterns but suffers when SPARQL language features such as FILTER, LIMIT are used.

3.3.4 Data Semantic Preservation

This solution takes an existing RDB as input, and extracts its metadata representation (MTRDB). Based on the MTRDB, a Canonical Data Model (CDM) is generated. Finally, the structure of the classification scheme in the CDM model is converted into OWL ontology and the recordsets of database are stored in owl document. A prototype has been implemented, which migrates a RDB into OWL structure, for demonstrate the practical applicability of our approach by showing how the results of reasoning of this technique can help improve the Web systems.

3.3.5 DartGrid Semantic

Wu et al. and Chen et al. describe the Dartgrid Semantic Web toolkit that offers tools for the mapping und querying of RDF generated from RDB. The mapping is basically a manual table to class mapping where the user is provided with a visual tool to define the mappings. The mappings are then stored and used for the conversion. The construction of SPARQL queries is assisted by the visual tool and the queries are translated to SQL queries based on the previously defined mappings. A full-text search is also provided.

The tool is available at: http://ccnt.zju.edu.cn/projects/dartgrid/

3.3.6 Asio Semantic Bridge for Relational Databases (SBDR) and Automapper

Asio Semantic Bridge for Relational Database and Automapper: Asio Semantic Bridge for Relational Databases (SBRD) and Automapper use the table to class approach. Automapper generates an OWL Full ontology from a relational database. In the generated ontology, each class corresponds to a table in the relational database and columns are represented as properties of the relevant class. A primary key column has cardinality set to 1. A nullable column has max cardinality set to 1. For a foreign key, an object property is created and its range is set to the corresponding class. The generated ontology includes SWRL rules to equate individuals based on multiple primary key columns. Semantic Bridge for Relational Databases provides an RDF view of data in the relational database. SPARQL queries can be written in terms of the Automapper generated data source ontology and relational data is returned as RDF SBRD rewrites the SPARQL query to SQL, executes the SQL and converts SQL rows to RDF conforming to the data source ontology.

3.3.7 XTR-RTO

In XTR-RTO, eXtensible Markup Language (XML) document is used as source for generating OWL ontology. The translations process converts XML schema to RDB schema and then, RDB schema to OWL ontology. Different attributes used to describe RDB: DbName, Relation, RelationList, Table and Attribute. Tables are converted to rdb: Relation and rdb: RelationList, and each attribute is mapped to rdf: Attribute and rdb: hasType.The entityrelation model is used for organizing database, which clearly expresses the relationship between data. Therefore, metadata information and structural restrictions extracted from relational database to construct ontologies.

Mapping approach used in XTR-RTO:

- Each table is converted to rdb: Relation and each attribute is mapped to rdb:Attribute and rdb:hasType
- Rdb:ReferenceAttribute and rdb:ReferenceRelation are generated for foreign key attribute.

There is one table in this relational database, as illustrated in Table II

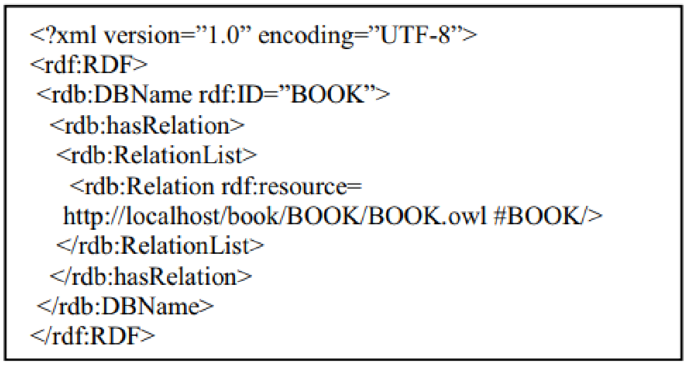
BOOK_ID	TITLE	AUTHOR	PRINTER_ID

Figure 2 BOOK TABLE IN RELATIONAL MODEL

Suppose database is created on local computer, and the OWL ontology has a namespace

“http://mylocalpc/book.owl”. Fig. 2 illustrates the relations

in the ontology:



```
<?xml version=”1.0” encoding=”UTF-8”>
<rdf:RDF>
 <rdb:DBName rdf:ID=”BOOK”>
   <rdb:hasRelation>
    <rdb:RelationList>
      <rdb:Relation rdf:resource=
     http://localhost/book/BOOK/BOOK.owl #BOOK/>
    </rdb:RelationList>
   </rdb:hasRelation>
 </rdb:DBName>
</rdf:RDF>
```

Figure 3 Relations in the ontology
The ontology generated from table BOOK is shown in the Fig.4

```
<rdb:Relation rdf:ID=”BOOK”>
 <rdb:hasAttribute>
  <rdb:AttributeList>
    <rdb:Attributerdf:resource=
”http://localhost/book/BOOK/BOOK.owl#BOOK_ID”/>
    <rdb:Attribute rdf:resource=
”http://localhost/book/BOOK/BOOK.owl#PRINTER_ID”/
>
  </rdb:AttributeList>
 </rdb:hasAttribute>
</rdb:Relation>
<rdb:PrimarKey rdf:ID=”BOOK_ID”>
<rdb:isNullable>false</rdb:isNullable>
<rdb:type>string</rdb:type>
</rdb:PrimarKey>
<rdb:Attribute rdf:ID=”PRINTER_ID”>
            <rdb:isNullable>false</rdb:isNullable>
            <rdb:type>string</rdb:type>
</rdb:Attribute>
<rdb:referenceAttribute>
            <rdf:Attribute rdf:resource=
            http://localhost/book/PRINTER/PRINTER.owl
            #PRINTER#PRINTER_ID/>
</rdb:referenceAttribute>
```

Figure 4 OWL description of BOOK table

3.3.8 RTAXON

A data mining approach for ontology construction. This technique mines (extracts) database contents to find categorization patterns (subsumption relation). These categorization patterns are then used to generate class hierarchies. Concept hierarchy identification by identifying the relations found in attributes of the relation is used as a base for learning ontology. Due to the assumption that attribute names tell meaning full role in the table/relation, lexical clues of attribute names are identified (i.e. for categorizing the tuples). RDBToOnto is a tool that uses RTAXON method for converting RDB to ontology.

3.3.9 Ontology Generator (RDB2On)

The ontology generator, RDB2ON, is an automatic method, to spare the time. RD2ON is created on the base of eight mapping rules. Ontology generator is partitioned into three modules: (1) Data base dissection: in this module, reverse engineering is applied to extract the schema of an RDB. The data like tables, columns, primary keys, foreign keys and so on is procured. This data is then assembled to apply the rules explained before. (2) Schema transformation: conversion of Rdb to owl ontology is carried out in this module. (3) Owl ontology generation: the last ontology report is created in the third module.

3.3.10 RDBToOnto

"RDBtoONTO" is semiautomatic tool for generation of ontology. The methodology helps the client to get a populated ontology. Cebrah insists on normalization of RDB for ontology generation. Normalization guarantees disposal of information duplications. Mining concepts are likewise applied in RDBtoONTO tool. But mapping rules are not talked over, and the tool implementation is not discussed.

An alternate issue with few executed models like, mapping master, RDBtoONTO and "relational-owl" is that they are not effectively accessible.

3.3.11OntoEdit

OntoEdit, professional version, is an ontology engineering environment to support the development and maintenance of an ontology. Ontology development process in OntoEdit is based on their own methodology, On-To-Knowledge discussed in 1.1(d) which is originally based on Common KADS. Two tools, OntoKick and Mind2Onto, are prepared for supporting the phase of ontology capture. OntoKick is designed for computer engineers who are familiar with software development process and tries to build relevant structures for building informal ontology description by obtaining competency questions discussed in 1.1(b) which the resulting ontology and ontology-based applications have to answer. Mind2Onto is a graphical tool for capturing informal relations between concepts. It is easy to use because it has a good visual interface and allows loose identification of relations between concepts. However, it is necessary to convert the map into a more formal organization to generate an ontology.

The refinement phase is for developers to use an editor to refine the ontological structure and the definition of concepts and relations. Like most of other tools, OntoEdit employs the client/server architecture where ontologies are managed in a server and multiple clients access and modify one. A sophisticated transaction control is introduced to enable concurrent development of an ontology in a collaborative manner. It employs Ontoclean method mentioned in 2.2.2 and discussed in Part 3 to refine the is-a hierarchy. The key process in the evaluation phase is use of competency questions obtained in the first phase to see if the designed ontology satisfies the requirements. To do this, OntoEdit provides users with a function to form a set of instances and axioms used as a test set for evaluating the ontology against the competency questions. It also provides users with debugging tools for ease of identify and correct incorrect part of the ontology. It maintains the dependency between competency questions and concepts derived from them to facilitate the debugging process. This allows users to trace back to the origins of each concept. Another unique feature of this phase is that collaborative evaluation is also supported by introducing the name space so that the inference engine can process each of test sets given by multiple users. OntoEdit employs F-Logic as its inference engine. It is used to process axioms in the refinement and evaluation phases. Especially, it plays an important role in the evaluation phase because it processes competency questions to the ontology to prove that it satisfies them. It exploits the strength of F-logic in that it can express arbitrary powerful rules which quantify over the set of classes which Description logics cannot.

3.3.12WebODE

WebODE is a scalable and integrated workbench for ontology engineering based on the ontology development methodology METHONTOLOGY described in 1.1(c). It supports building an ontology at the knowledge level, and translates it into different ontology languages. WebODE is designed on the basis of a general architecture shown in Fig. 3 and covers most of the processes appearing in the ontology lifecycle. While Protégé-2000 and OntoEdit are based on plug-in architecture, WebODE is based on a client-server architecture which provides high extensibility and usability by allowing the addition of new services and the use of existing services. Ontologies are stored in an SQL database to attain high performance in the case of a large ontology.

It has export and import services from and into XML, and its translation services into and from various ontology specification languages such as RDF(S), OIL, DAML+OIL, X-CARIN, Jess and F-Logic. Like OntoEdit, WebODE's ontology editor allows the collaborative edition of ontologies. Although WebODE is an integrated tool sets covering most of the activities in ontology lifecycle, it has no explicit stepwise guidance function unlike Hozo.

In the ontology development phase, WebODE has ontology editing service, WAB: WebODE Axiom builder service, inference engine service, interoperability service and ontology documentation service. The ontology editor provides users with form based and graphical user interfaces, WAB provides an easy graphical interface for defining axioms. It enables users to define an axiom by using templates given by the tool with simple mouse operations. Axioms are translated into Prolog. The inference engine is based on Prolog and OKBC protocol[http://www.ai.sri.com/~okbc/] to make it implementation-independent. Interoperability services provided by WebODE are of variety. It includes ontology access API, ontology export/import in XML-family languages, translation of classes into Java beans to enable Jess system to read them and OKBC compliance.

The collaborative editing of an ontology is supported by a mechanism that allows users to establish the type of access of the ontologies developed through the notion of groups of users. Synchronization mechanism is also introduced to enable several users to safely edit the same ontology. To support the use process of ontology, WebODE has several functions. Like Hozo, WebODE allows users to have multiple sets of instances for an ontology by introducing instance sets depending on different scenarios, and conceptual views from the same conceptual model, which allows creating and storing different parts of the ontology, highlighting and/or customizing the visualization of the ontology for each user. WebPicker is a set of wrappers to enable users to bring classification of products in the e-Commerce world into WebODE ontology. ODEMerge is a module for merging ontologies with the help of correspondence information given by the user.

3.3.13 Protégé-2000

Protégé-2000 is strong in the use phase of ontology: Use for knowledge acquisition, merging and alignment of existing ontologies, and plug-in new functional modules to augment its usability. It has been used for many years for knowledge acquisition of domain knowledge and for domain ontology building in recent years. Its main features include:

(1) Extensible knowledge model to enable users to redefine the representational primitives.

(2) A customizable output file format to adapt any formal language

(3) A customizable user interface

(4) Powerful plug-in architecture to enable integration with other applications

These features make Protégé-2000 a meta-tool for domain model building, since a user can easily adapt it to his/her own instance acquisition tool together with the customized interface. It is highly extensible thanks to its very sophisticated plugin architecture. Unlike the other three, Protégé-2000 assumes local installation rather than use through internet using client/server architecture. Its knowledge model is based on frame similar to other environments. Especially, the fact that Protégé-2000 generates its output in many ontology languages and its powerful customizability make it easy for users to change it to an editor of a specific language. So-called "meta-tuning" can be easily done thanks to Protégé's declarative definition of all the meta-classes which play a role of a template of a class. Protégé has a semi-automatic tool for ontology merging and alignment named PROMPT discussed in 2.3.2. It performs some tasks automatically and guides the user in performing other tasks.

3.3.14OE: Ontology editor in Hozo

“Hozo1 ” is an integrated ontology engineering environment for building/using task ontology and domain ontology based on fundamental ontological theories. “Hozo” is composed of “Ontology Editor”, “Onto-Studio” and “Ontology Server”. The ontology and the resulting model are available in different formats (Lisp, Text, XML/DTD, DAML+OIL) that make it portable and reusable. One of the most remarkable features of Hozo is that it can treat the concept of Role. When an ontology is seriously used to model the real world by generating instances and then connecting them, users have to be careful not to confuse the Role such as teacher, food, fuel, etc. with other basic concepts such as human, vegetable, oil, etc. The former is a role played by the latter. For example, if one builds an ontology including and , then when he quits the teacher job, he cannot be an instance of the class of teacher, and hence he cannot be an instance of the class human, which means he must die. This difficulty is caused by making an instance of Role which cannot have an instance in theory. In Hozo, three different classes are introduced to deal with the concept of role appropriately.

Role-concept: A concept representing a role dependent on a context(e.g., teacher role)

Basic concept: A concept which does not need other concepts for being defined(e.g., human)

Role holder: An entity of a basic concept which is holding the role(e.g., teacher)

A basic concept is used as the class constraint. Then an instance that satisfies the class constraint plays the role and becomes a role holder. Hozo supports to define such a role concept as well as a basic concept. In each step Onto-Studio, which supports AFM method described in 1.1(e), provides users with graphical interfaces to help them perform the suggested procedures. The output of Onto-Studio is a rather informal representation of ontology which is in turn translated by the system into the Ontology editor representation to enable users to define ontology more rigorously.

Like other editors, Ontology Editor in Hozo provides users with a graphical interface through which they can browse and modify ontologies by simple mouse operations. Users do not have to worry about so-called coding to develop an ontology. The internal representation of the ontology editor, which is hidden from users, is XML and it generates DAML+OIL code to export the ontology and instance. It treats “role concept” and “relation” on the basis of fundamental consideration discussed. This interface consists of the following four parts:

1. Is-a hierarchy browser displays the ontology in a hierarchical structure according to only is-a relation between concepts.

2. Edit panel is composed of a browsing panel and a definition panel. The former displays the concept graphically, and the latter allows users to define the selected concept in the is-a hierarchy browser.

3. Menu bar is used for selecting tools

4. Tool bar is used for selecting commands

Collaborative development of an ontology is supported in the Ontology Editor. At the primitive level, the ontology server allows users to read and copy all the ontologies and instances, but do not allow modification of them by users other than the original developer of them. Thus, unlike OntoEdit and WebODE, Hozo does not allow multiple users to edit the same concept at the same time. Instead, Ontology Editor allows users to divide an ontology into several component ontologies and manages the dependency between them to enable the concurrent development of an ontology. The dependency between the component ontologies are three fold: super-sub relation(is-a relation), referred-to relation(class constraint) and task-domain relation. In the current implementation, the first two are taken into account. The system observes every change in each component ontology and notifies it to the appropriate users who are editing the ontology which might be influenced by the change. The notification is done based on the 16 patterns of influence propagation analyzed beforehand. The notified users can select a countermeasure among the three alternatives: (1)to adapt his/her ontology to the change, (2)not to adapt to the change but stay compliant with the last version of the changed ontology and (3)neglect the change by copying the last version into his/her ontology. The timing of the notification is selected by the users among the two: when the editing task has been initiated and he/she requested.

Functionality and GUI of Hozo's instance editor is the same as the one for ontology. The consistency of all the instances with the ontology is automatically guaranteed, since a user is given valid classes and their slot value restrictions by the editor when he/she creates an instance. Hozo has an experience in modeling of a real-scale Oil-refinery plant with about 2000 instances including even pipes and their topological configuration which is consistent with the Oil-refinery plant ontology developed with the help of domain experts. The model as well as the ontology are served by the ontology server and can answer questions on the topological structure of the plant, the name of each device, etc. Any ontology built by Hozo can have multiple sets of instances which are independent of one another.

The ontology server stores ontologies and instance models in an XML format and serves them to clients through API compliant with OKBC protocol. Ontology editor is also a client of the ontology server. Inference mechanism of Hozo is not very sophisticated. Axioms are defined for each class but it works as semantic constraint checker like WebODE.

3.3.15RDB2RDF

A. Transformation Approaches

Approach	**Task**	**Ontology exists?**	**Query, Dump**	**Usability**	**OWL Schema**	**Schema/Data Separation**
CODE (Man Li)	conversion	ad-hoc creation	dump/batch, ? with possibility to update instance data?	semi-automatic, but clear engineering process	automatic learning of some hierarchy, domain/range of roles, object properties, 11 rules	yes, separate, data is imported after schema creation
D2RQ	uni-directional, SPARQL, LinkedData	ad-hoc creation	query, dump possible	automatic mapping	automatic mapping generates poor schema, one table per class, only datatype properties, more possible	no, everything created at once
DartGrid	integration of several DBs	integration ontology mandatory	query			

DB2OWL	several DBs, aims at integration	distinction between a global ontology and local ontologies , existence of a global target ontology mandatory	query, uses R2O	automatic	basically same as CODE (Man Li), similar rules	yes
Habegger	learning of a mapping	no	n.a.	proposal of a learning algorithm, no evaluation	some hierarchy based on theta subsumption in horn clauses, object properties	??
OntER(Trinkunas)	verify acquisition	no	manual acquisition of the ontology based on ER model	manual, but with clear directives, ?could be automated?	conversion of ER model, similarity to RDF is exploited	no, but a verification method seems to exist, to validate the handcrafted model against the DB, based on a DB integration tool
OpenLink Virtuoso RDF Views	uni-directional, SPARQL, LinkedData, simultaneously access multiple remote or local DBs via SPARQL	ad hoc creation	query, dump possible	Mapping freely specifiable. Embedded SQL/SPARQL conditions for arbitrary mappings.		Mapping stored in database as with SQL Views.
Protege DataGenie						
Protege DataMaster						
Protege OntoBase						
R2O+ODEMapster						
Relational.OWL						
SPASQL						
SquirrelRDF	uni-directional, access several DBs via SPARQL	Config is (augmented) rdfs	query, allows SPARQL	automatic mapping	no, on purpose not	no
Triplify	uni-directional, Linked Data	ad-hoc creation	display of LinkedData	manual SQL queries	supports classes and object properties, early version	no, everything created at once

B. List of Available Tools

1. D2RQ
2. Triplify
3. SquirrelRDF
4. Relational.OWL
5. Protege plugins
   - DataMaster
   - OntoBase
   - DataGenie

6. Virtuoso's RDBMS to RDF Meta Schema Language
7. **The Semantic Discovery System**: Provides the functionality to rapidly build solutions for non technical Users to create and execute Ad Hoc queries using the network Graph User Interface (SPARQL to SQL is auto generated). Integrates and interconnects ALL data silo types - providing a virtual Semantic Web interface to all RDBMS's, Web Services, Excel Spreadsheets, and any Hybrid File Systems.
   - Semantic Discovery Systems (Main Site)
   - Semantic Discovery Systems ("Trailer"/Summary Web site)
8. lgraps mentioned in the mailinglist (out of scope)
9. Protege Excel_Import(out of scope)
10. R2D2
11. dbview
12. Amara

4 Conclusion

Many techniques and tools have been proposed over the last years to enable the publication of relational data on the web in RDF. In this document, we have described it is possible to categorize such RDB-to-RDF approaches. Based on these axes, we have proposed a review of many of RDB-to-RDF tools. The categorization proposed helped us identify commonalities and differences between existing approaches, as well as the correlations between the solutions applicable to each step of the RDB-toRDF translation process. We observed that producing RDF data with sufficiently rich semantics is a critical concern of most approaches studied, in order to make the data usable, interoperable and linkable. Therefore, we have briefly presented various strategies investigated in the literature to produce richer semantic data.

5 References

[1] A survey of RDB to RDF translation approaches and tools, Franck Michel, Johan Montagnat, Catherine Faron Zucker

[2] Automapper: Relational Database Semantic Translation using OWL and SWRL, Matthew Fisher and Mike Dean

[3] A Survey of Current Approaches for Mapping of Relational Databases to RDF, W3C RDB2RDF Incubator Group January 08 2009

[4] Requirements for Relational to RDF Mapping, Orri Erling