Publishing Linked Data Using web2py

Chris Baron* and Massimo Di Pierro†
School of Computing, DePaul University, Chicago, IL

October 28, 2010

Abstract

In this paper we describe a software plugin to publish any database as Linked Data using the web2py Database Abstraction Layer. It works with new database tables as well as with existing ones. Both simple and complex RDF structures can be expressed with ease and the plugin automatically creates web services to expose the data. The Database Abstraction Layer writes the SQL dyanamically and transparently. It supports ten different database back-ends, including Oracle and Google Big Table. This approach leverages on the capabilities of the framework, including a Role Based Access Control system with pluggable authentication methods. Our system runs on any platform supported by Python including the Google App Engine cloud platform. We believe our approach can contribute to make programming for the semantic web more accessible.

1 Introduction

The amount of information posted online is rapidly growing beyond the point where humans can comprehend it and we will need automated agents to be able to find, collect, and summarize information for us. Most of this information already exists in structured form, in relational databases.

It has been shown that the growth of traditional Web pages supersedes the growth of semantic representations and most web developers are unaware of the Semantic Web. As a result, a significant portion of Web data is unavailable to semantic search engines. Despite the fact that there are many tools already available to program for the Semantic Web we believe there are not enough tools that make it easy. This is the problem we try to address in this paper.

Modern web applications rarely use raw SQL to communicate with databases but, instead, they use a Object Relational Mapper (ORM) or a Database Abstraction Layer (DAL) to map programming objects into database objects. The DAL write the SQL dynamically in the specific dialect of the database back-end.

*topher.baron@gmail.com
†mdipierro@cs.depaul.edu

This approach ensures portability of the application over multiple databases and prevents SQL Injection attacks.

The purpose of this paper is to show how we can take advantage of one of these Web Frameworks and its DAL (specifically we will use WEB2PY [3]) to annotate database tables, fields and relations with OWL and expose the data in RDF. The proposed system is powerful enough to translate one-to-many and many-to-many database relations into both simple and complex RDF expressions and expose them via a RESTful web service.

At the time of writing, the WEB2PY DAL supports transparently SQLite, MySQL, PostgreSQL, Oracle, DB2, Informix, FireBird, Ingres, MSSQL and Google Big Table (the non-relational database provided by Google App Engine).

Using the technology described here any of the relational database listed above can became a node of the semantic web. Not all tables and records will be exposed in RDF, only those appropriately annotated.

WEB2PY also includes a Role Based Access Control mechnism that can be enforced on any object.

We believe this mechanism simplifies the process of exposing data in RDF.

2 Linked Data

Semantic Web is a group of methods and technologies to allow machines to understand the meaning - or "semantics" - of information on the World Wide Web.

The Web Ontology Language (OWL) is a collection of knowledge representation languages for describing ontologies. These languages are characterised by formal semantics often expressed in RDF (and XML-based serializations).

The World Wide Web Consortium (W3C) created a Semantic Web Education and Outreach (SWEO) interest group that expired in 2008 [4]. The group started the Linking Open Data project with the goal of building a data commons for "making various open data sources available on the Web as RDF and by setting RDF links between data items from different data sources". According to the W3C, over 4.7 billion RDF triples have been published to date, and they are connected by about 142 million RDF links [5].

Examples of Linked Data Ontologies are:

- dc: Dublin Core. A set of text elements to catalog generic online resources [6].
- sioc: Semantically-Interlinked Online Communities ontology, which aims to describe relation between online communities [7].
- foaf: Friend-of-a-Friend ontology which describes relations between people [8].

Here is a basic example of how to describe a blog entry using SIOC:

```
<sioc:Post rdf:about="http://...">
    <dcterms:title>Creating connections between ...</dcterms:title>
    <dcterms:created>2006-09-07T09:33:30Z</dcterms:created>
    <sioc:has_container rdf:resource="http://..."/>
    <sioc:has_creator>
        <sioc:User rdf:about="http://...">
            <rdfs:seeAlso rdf:resource="http://..."/>
        </sioc:User>
    </sioc:has_creator>
    <sioc:content>SIOC provides a .... </sioc:content>
    <sioc:topic rdfs:label="Semantic Web" rdf:resource="http://..."/>
</sioc:Post>
```

Notice how every resource is dentified by a URL and how the tag name and attributes specify the relations between resources.

3 web2py

Web Frameworks are collections of libraries that abstract many of the basic tasks of web applications such as parsing cookies, managing sessions, handling persistence, dispatching HTTP requests and mapping them into function calls, validating input for security, logging errors, pooling database connections for speed, abstracting details about the database back-end, and more. There are Web Frameworks written in almost every programming language.

WEB2PY is one of these web frameworks, one of the few born in an academic environment. WEB2PY is written in the Python programming language and it is programmable in Python. It is designed with three primary goals: be easy to use, force developers to follow good software engineering practices, provide strong security (prevents SQL Injections and XSS vulnerabilities).

WEB2PY does not require installation and includes a web server with SSL capabilities, the SQL(ite) relational database, a web based IDE (with editor, testing and debugging capabilities), a template language similar to PHP, a Model View Controller design, a Database Abstraction Layer that writes the SQL for you (it also CREATE, and ALTER tables as necessary), a web based administrative interface for the database, a Role Based Access Control system with pluggable authentication methods, a mechanism for automatic generation of forms with validation of all form fields, a plugin system, and an internationalization system. It works with multiple web servers (for example Apache) and databases. It is the only framework that allows you to deploy apps on the Google Cloud without the need to program in the native Google API.

In WEB2PY an application consists of files that are divided in the following basic groups[1]:

- **Models**: These are files that contain a description of the data representation using the Database Abstraction Layer. Typically a Model contains a definition of a set of tables and their relations. WEB2PY generates all

[1] An application may also contain static files, modules, plugins, private files, session files, cache files, error logs, and translation files to deal with internationalization.

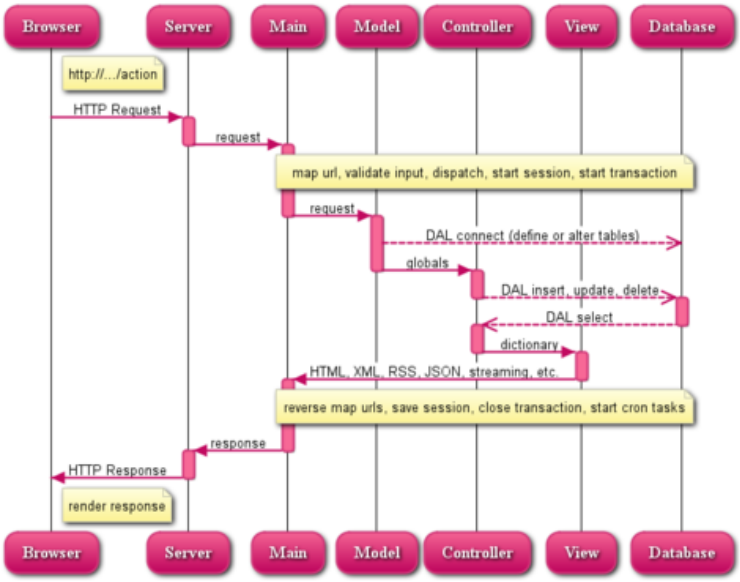

Figure 1: Information flow inside the WEB2PY framework. The proposed annotation mechanism resides in the Model and exposes the Linked Data as a RESTful web service via the provided controller `plugin_rdf.py`.

required SQL to create the tables if they do not exist, alter them if needed, and query them. It also automatically creates a web based interface to the database from models.

- **Controllers**: They contain the logic and control flow of the application. A controller may contain one or more functions called "actions". The framework maps URLs into actions.

- **Views**: They describe how to serialize the data returned by the actions, for example in HTML, XML, JSON, CSS, RSS, and/or RDF.

Each group of files is located in its own folder under the application folder. When the web server receives a request from a client, it runs the models, then maps the URL of the HTTP request into an action, runs the action, renders the output of the action into - for example - HTML using the associated view for the specific requested format. The resulting HTML is returned to the requesting client.

The information flow within the framework is graphically depicted in fig. 1.

For the purpose of this paper we only need to understand the syntax of Model files. That is where all the RDF annotation will be done. In order to expose the RDF data as a service we created a single controller file called "plugin_rdf.py". This file does not need to be edited, it just needs to be dropped in the controllers folder for the application or installed as a plugin and, for practical purposes, it can be treated as a black-box that extends WEB2PY functionality.

We refer to the official WEB2PY documentation for further details [3].

From now on we will assume we have a WEB2PY sever running on localhost (127.0.0.1) on port 8000 and we are working on an application called "semantic". The content of this application includes exclusively default files created by WEB2PY and the files described below.

Some screenshots for WEB2PY are shown in fig. 2.

4 Annotating the Relational Database

In this section we provide a practical example of how to create a model, annotate its tables, fields and relations with OWL, and expose it via a RESTful RDF service.

In the following we will adopt the convention that table names are singular.

The example we consider is a Blogging application that defines the following tables:

- **Post**: It stores each blog post's text content, its author, and its title.

- **Comment**: A comment is associated with an author and the body of the comment.

- **Tag**: It is simply a collection of keywords, not to be confused with RDF tags. These tags (for example "Obama", "Politics", "Democrats") are

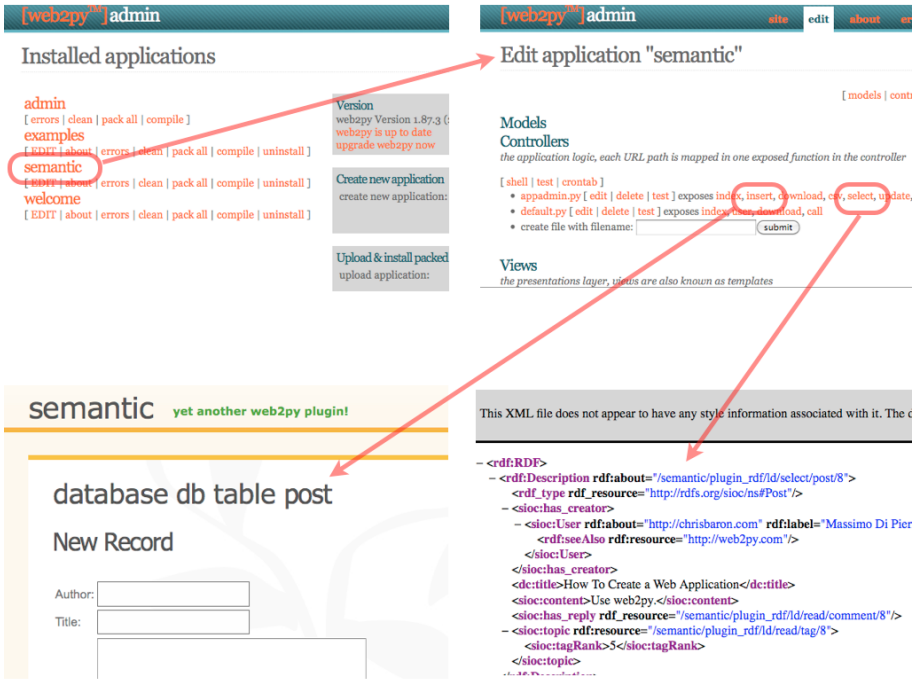

Figure 2: The top-left image shows the main page of tha web based IDE, it lists installed applications. The top-right page is also part of the web based IDE and it lists the content of our example application. The bottom-left image shows the database administrative internface for our example DB. The bottom-right image shows an example Linked Data output as rendered by Firefox.

associated with blog posts as a quick categorization tool: blog readers can find posts tagged with topics they are interested in.

- **Post_Tag_Link**: A post may have multiple tags and a tag may be associated to multiple posts. This table implements such linkage. The link table also has a ranking field since some tags apply to a certain post more than other posts.

The relation between posts and Comments is one-to-many. The relation between posts and tags, as provided by the post_tag_link table is many-to-many.

This is how the aforementioned tables are defined in a WEB2PY model (which we will call "models/db_example.py"):

```
db.define_table('post',
                Field('author', 'string'),
                Field('title', 'string'),
                Field('body', 'text'))

db.define_table('comment',
                Field('post_id', db.post),
                Field('author', 'string'),
                Field('body', 'text'))

db.define_table('tag',
                Field('name', 'string'))

db.define_table('post_tag_link',
                Field('post_id', db.post),
                Field('tag_id', db.tag),
                Field('ranking', 'integer'))
```

Here `db` represents a database connection object. Each table has a name and fields. The first argument of a Field is the field name, the second argument is the field type. When the field type is another table than it is a foreign key reference field. Once a table is defined, WEB2PY creates the table if it does not exist or alters it as necessary.

Until this point we have done nothing specific about the Semantic Web yet. In order to expose the data in these tables as RDF we annotate them with OWL. This is done in 3 steps.

4.1 Step 1: exposing the service

The first step consists of installing the following plugin into the application:

```
web2py.plugin.rdf.w2p
```

from

```
http://web2py.com/examples/static/web2py.plugin.rdf.w2p
```

This is done from the administrative interface of web2py (at the bottom of the top-right screenshot in fig. 2). The plugin contains a controller file "plugin_rdf.py". This will create and expose all required services which can be accessed via a RESTful interface:

```
http://127.0.0.1:8000/semantic/plugin_rdf/ld/tables
http://127.0.0.1:8000/semantic/plugin_rdf/ld/select/<table>
http://127.0.0.1:8000/semantic/plugin_rdf/ld/read/<table>/<record>
```

Here "semantic" is the application name; "plugin_rdf" is the plugin controller; "ld" is the Linked Data action defined by the controller; "table", "select" and "read" are the functions performed by the action; `<table>` and `<record>` refer to a table annotation as explained below.

4.2 Step 2: annotating fields

The second step consists of annotating the individual fields of each table. We annotate the fields of the `db.post` table with ontologies by adding the following code immediately after the table definition:

```
db.post.author.rdf = 'sioc:has_creator'
db.post.title.rdf = 'dc:title'
db.post.body.rdf = 'sioc:content'
```

The left-hand-side of each expression refers to the optional RDF attribute of a table column. The right-hand-side is the OWL property that the column refers to . The prefix refers to the ontology namespace: "sioc:" and "dc" respectively.

4.3 Step 3: annotating tables

So far we have described how records in the `db.post` table are to be represented in RDF. Here we describe what a `db.post` is in OWL, by defining the `db.post` table's "rdf" property which is implemented as a Python dictionary of dictionaries. After the previous code, we append:

```
db.post.rdf = {
    'type': 'http://rdfs.org/sioc/ns#Post',
    'namespaces': {
         '_xmlns:dc':'http://purl.org/dc/elements/1.1/',
         '_xmlns:sioc':'http://rdfs.org/sioc/ns#'
    },
    'references': {
         'comment': 'sioc:has_reply'
    }
}
```

The top level dictionary keys have the following meaning:

- The "type" key of the dictionary defines what a `db.post` is in the Semantic Web. In our case, the SIOC ontology has defined a "Post" class which all of our posts are an instance of.
- "namespaces" defines the OWL namespaces to be included in order to properly serialize a `db.post` in RDF/XML. Recall in the previous step, annotating fields, we defined a post's author to be associated with the `sioc:has_creator` property, the title with `dc:title`, and body as `sioc:content`. This means we need to include namespaces for sioc and dc.

- The "references" key defines how matching records from foreign key tables are exposed. Here we are saying that when a `db.post` is serialized, the `sioc:has_reply` property will be used to reference all related comments.

In this paper we refer to code written in steps 2 and 3 as RDF annotations.

4.4 More annotations

Similarly we can annotate other tables (tag and comment):

```
db.comment.author.rdf = 'sioc:has_creator'
db.comment.body.rdf = 'sioc:content'

db.comment.rdf = {
    'type': 'http://rdfs.org/sioc/ns#Post',
    'namespaces': { "_xmlns:sioc":"http://rdfs.org/sioc/ns#" }
}

db.tag.name.rdf = 'sioc:content'

db.tag.rdf = {
    'type': 'http://rdfs.org/sioc/ns#topic',
    'namespaces': { "_xmlns:sioc":"http://rdfs.org/sioc/ns#" }
}
```

4.5 Exposing the Service

After the database tables are annotated, the web services are exposed automatically by web2py.

Visit the following URL:

```
http://127.0.0.1:8000/semantic/plugin_rdf/ld/tables
```

to get a semantic description of our database:

```
<rdf:RDF>
  <rdf:Description rdf:about="/semantic/plugin_rdf/ld/tables">
    <rdf:type rdf:resource="http://www.dbs.cs.uni-duesseldorf.de/RDF/
        relational.owl#Database"/>
    <relational:hasTable
     rdf:resource="/semantic/plugin_rdf/ld/select/post"/>
    <relational:hasTable
     rdf:resource="/semantic/plugin_rdf/ld/select/comment"/>
    <relational:hasTable
     rdf:resource="/semantic/plugin_rdf/ld/select/tag"/>
  </rdf:Description>
</rdf:RDF>
```

Specifically, the url exposes a Linked Data resource listing database table resources [25]. WEB2PY looks for all tables with an rdf property as a filter for this resource.

Looking at the post table:

```
http://127.0.0.1:8000/semantic/plugin_rdf/ld/select/post
```

we receive information about the table and its records:

```
<rdf:RDF>
  <rdf:Description rdf:about="/semantic/plugin_rdf/ld/select/post">
    <rdf:type rdf:resource="http://www.dbs.cs.uni-duesseldorf.de/RDF/
        relational.owl#Table"/>
    <relational:has
     rdf_resource="/semantic/plugin_rdf/ld/read/post/1"/>
    <relational:has
     rdf_resource="/semantic/plugin_rdf/ld/read/post/2"/>
  </rdf:Description>
</rdf:RDF>
```

Our post is an instance of the relational OWL class Table and it contains two rows.

Continuing on to one of the records:

```
http://127.0.0.1:8000/semantic/plugin_rdf/ld/read/post/1
```

```
<rdf:RDF>
  <rdf:Description rdf:about="/semantic/plugin_rdf/ld/select/post/1">
    <rdf_type rdf_resource="http://rdfs.org/sioc/ns#Post"/>
    <sioc:has_creator>Chris Baron</sioc:has_creator>
    <dc:title>How To Publish RDF Using web2py</dc:title>
    <sioc:content>
       How To Publish RDF Using web2py, really
    </sioc:content>
    <sioc:has_reply
     rdf_resource="/semantic/plugin_rdf/ld/read/comment/1"/>
    <sioc:topic
     rdf:resource="/semantic/plugin_rdf/ld/read/tag/1">
      <sioc:tagRank>10</sioc:tagRank>
    </sioc:topic>
  </rdf:Description>
</rdf:RDF>
```

This is a `db.post` defined semantically. Notice the `rdf:type` element, it indicates that the post is an instance of the type defined in Ref.[7], which we set via the `type` attribute of the `db.post.rdf`. Each of the columns with an rdf attribute are transformed into their associated ontology where the record values are literals inside these XML elements. The `sioc:has_reply` is not referenced as a column's rdf attribute, but was retrieved from the "references" property of the `db.post.rdf` data structure.

web2py also recogizes many-to-many relations implemented via link tables (as in post tag link). Link tables are special because their records do not just connect two or more tables. They may also contain additional information associated to the link (like for example the ranking in the case of post tag link). We have the option to propagate this information and include in the requested entries connected by the link. Here is an example:

```
db.post_tag_link.rdf_mapping =
    { 'post': { 'reference': 'sioc:topic',
                'columns': { 'ranking': 'sioc:tagRank' } } }
```

Note that if we remove the "columns" attribute from the above code, the post serialization is the same except for the `sioc:tagRank` element disappears.

Also, if the post attribute is removed, the `sioc:topic` object property is replaced by the default property `relational:references`.

4.6 Complex annotations

While the previous example covers many cases of practical interests, in some cases it may not be sufficient and further customization may be necessary. Here is a more complex example of how one may annotate the `db.post.author` field:

```
db.post.author.rdf = {
    'name':'sioc:has_creator',
    'children': [ {
        'name':'sioc:User',
        '_rdf:about':'http://chrisbaron.com',
        '_rdf:label':'$VALUE',
        'children': [ { 'name':'rdf:seeAlso',
                        '_rdf:resource':'http://web2py.com'
                    } ]
    } ]
}
```

Above, `$VALUE` is replaced with the database record's value, the "children" attribute denotes child nodes; "name" corresponds to the element name, and any attribute beginning with an underscore is an element attribute.

A request to

```
http://127.0.0.1:8000/semantic/plugin_rdf/ld/read/post/1
```

generates the following complex `sioc:hascreator` response:

```
<sioc:has_creator>
  <sioc:User rdf:about="http://chrisbaron.com" rdf:label="Chris Baron">
    <rdf:seeAlso rdf:resource="http://web2py.com"/>
  </sioc:User>
</sioc:has_creator>
```

5 Related Work

The idea of mapping Database Relations into RDF is not at all new. As already observed by [16], the Relational Model, which predates SQL, expresses the syntactic structure of the stored data and therefore it can be mapped into RDF. A number of authors have realized such mapping [9, 10, 11, 12, 13].

What is lacking in this approach is domain specific semantics, as expressed by OWL. In our system the domain specific semantic is injected by annotating the database tables, fields and relations in the DAL. This provides two advantages over other approaches: it is database agnostic and it is not domain specific.

Table 1 shows a comparative classification according to the W3C RDB2RDF Incubator Group as reported in ref. [24].

Approach	Auto(a)	Database(b)	Paradigm(c)	Map(d)	Domain(e)
Dartgrid [14]	Manual	Domain	SPARQL	Visual	specific
Hu et al. [15]	Auto	Both	ETL	intern	specific
Tirmizi et al. [16]	Auto	DB	ETL	FOL	general
Li et al. [17]	Semi	DB	ETL	n/a	general
DB2OWL [18]	Semi	DB	SPARQL	R2O	general
RDBToOnto [19]	Semi	DB+M	ETL	Visual	general
Sahoo et al. [20]	Manual	Domain	ETL	XSLT	specific
R2O [21]	Manual	DB+M	SPARQL	R2O	specific
D2RQ [22]	Auto	DB+M	LD,SPARQL	D2RQ	general
Virtuoso [23]	Semi	DB+M	SPARQL	own	general
Triplify [24]	Manual	Domain	LD	SQL	general
(this paper)	Semi	DB+M	LD	DAL	general

Table 1: Reference table of common mapping approaches. The classification criteria are: (a) degree of mapping creation automation: automatic, semi automatic, or manual; (b) database driven (DB), domain specific, or database driven with additional domain specific information (DB+M); (c) the resulting paradigm (LD for Linked Data); (d) language used for the mapping. (DAL refers to the web2py specific Database Abstraction Layer); (e) domain specific or general.

6 Conclusions

The web is a very young place, very much evolving, and it can offer us much more than we have experienced already. The next phase in its evolution is probably the semantic web. Today, web crawling is mostly achieved by parsing unstructured information from web sites (like Google does) and it has a high margin of error. Tomorrow, Linked Data users will be able to run programs (intellingent agents) that mine data in an intelligent way and generate the information by reasoning. Linked-data provide an extensible way to publish structured information.

In this paper we have shown a tool that allows to create a database (or connect to an existing database) using a Database Abstraction Layer and expose the data as RDF via a web service. While this idea is not new, our system allows to annotate arbitrary databases using ontological information in a portable way that is independent of the database back-end and can leverage on the capability of the WEB2PY framework for web application development.

Acknowledgements

References

[1] Berners-Lee, T.: Design Issues: Linked Data (2006), http://www.w3.org/DesignIssues/LinkedData.html

[2] Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web (2007), http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

[3] http://www.web2py.com/

[4] http://www.w3.org/2001/sw/sweo

[5] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/

[6] http://dublincore.org/documents/dcmi-terms

[7] http://rdfs.org/sioc/ns\#Post

[8] Brickley, D., Miller, L.: FOAF vocabulary specification 0.91. Technical report, ILRT (2007)

[9] de Laborda, C. P. and Conrad, S. 2005. Relational.OWL: a data and schema representation format based on OWL. In Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling

[10] Svihla M., Jelnek I. 2004. Two Layer Mapping from Database to RDF. In Proceedings of Electronic Computers and Informatics (ECI), Slovakia, Kosice

[11] Laclavik, M. 2006. RDB2Onto: Relational Database Data to Ontology Individual Mapping In: Tools for Acquisition, Organisation and Presenting of Information and Knowledge. P.Navrat et al. (Eds.),

[12] Bizer, C. Seaborne, A. 2004. D2RQ Treating Non-RDF Databases as Virtual RDF Graphs. Poster at 3rd International Semantic Web Conference (ISWC2004)

[13] Bizer, C. 2003. D2R MAP - A Database to RDF Mapping Language. The Twelfth International World Wide Web Conference (WWW2003)

[14] Zhaohui Wu, Shuming Tang, Shuiguang Deng, Jian Wu, Huajun Chen, Haijun Gao, "DartGrid II: A Semantic Grid Platform for ITS," IEEE Intelligent Systems, vol. 20, no. 3, pp. 12-15, May/June 2005, doi:10.1109/MIS.2005.44

[15] W. Hu and Y. Qu. Discovering simple mappings between relational database schemas and ontologies. In Proceedings of ISWC/ASWC2007, Busan, South Korea, volume 4825 of LNCS, pages 225238, 2007

[16] Sequeda, J F. Tirmizi, S.H. Corcho, O. Miranker, D.P. (2009) Direct Mapping SQL Databases to the Semantic Web. Technical Report 09-04. The University of Texas at Austin, Department of Computer Sciences.

[17] M. Li, X. Du, and S. Wang. A semi-automatic ontology acquisition method for the semantic web. In W. Fan, Z. Wu, and J. Yang, editors, WAIM, volume 3739 of LNCS, pages 209220. Springer, 2005.

[18] Cullot, N., Ghawi, R., Ytongnon, K.: DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. Universit de Bourgogne (2007)

[19] F. Cerbah. Learning highly structured semantic repositories from relational databases: The RDBToOnto tool. In Proc. of ESWC 2008, Tenerife, 2008.

[20] S. S. Sahoo, O. Bodenreider, J. L. Rutter, K. J. Skinner, and A. P. Sheth. An ontology-driven semantic mashup of gene and biological pathway information: Application to the domain of nicotine dependence. Journal of biomedical informatics, February 2008.

[21] Barrasa, J., Corcho, O., Gmez-Prez, A.: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372. Springer, Heidelberg (2005)

[22] C. Bizer and A. Seaborne, D2RQtreating non-RDF databases as virtual RDF graphs.. In: S.A. McIlraith, D. Plexousakis and F. van Harmelen, Editors, Proceedings of 3rd International Semantic Web Conference (ISWC04) Hiroshima, Japan. Springer, November (2004).

[23] Orri Erling and Ivan Mikhailov. RDF Support in the Virtuoso DBMS. In Sören Auer, Christian Bizer, Claudia Müller, and Anna V. Zhdanova, editors, CSSW, volume 113 of LNI, pages 5968. GI, 2007.

[24] Sören Auer *et al.*, Triplify Light-Weight Linked Data Publication from Relational Databases

[25] `http://www.schemaweb.info/schema/SchemaDetails.aspx?id=252`