# A Framework for OBDA: Current State and Perspectives

Sergio Alejandro Gómez[1,2] and Pablo Rubén Fillottrani[1,2]

[1]Laboratorio de I+D en Ingeniería de Software y Sistemas de Información (LISSI)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
San Andrés 800 - Campus Palihue – Bahía Blanca, Buenos Aires, Argentina
Email: {sag,prf}@cs.uns.edu.ar
[2]Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC-PBA)

**Abstract.** Ontology-based Data Access (OBDA) is concerned with providing end-users and applications with a way to query legacy databases through a high-level ontology that models both the business logic and the underlying data sources. The bridge between the ontology and the data sources is addressed by mappings that define how to express records of the database as ontological assertions. In this research, we are concerned with providing with tools for performing OBDA with relational and non-relational data sources. We developed a tool, which nowadays is in a prototypical state, that is able to access an H2 database, allowing the user to explicitly formulate mappings, and populating an ontology that can be saved for later querying. In this paper, we report on the advances we have made on the development of such a tool, which includes adding the functionality of creating, loading, saving a global ontology that can be populated with a database. Also the system allows the user to visually express mappings from the database to the ontology and the ability of creating databases for testing the behavior of the system in the presence of increasing workloads. The tests we performed indicate that the system is able to handle a moderate workload of tables of tens of thousands of records but fails to handle tables of millions of records.

**Keywords.** Ontology-based data access, Ontologies, Description Logics, Web Ontology Language, Relational databases.

## 1 Introduction

Ontology-based Data Access (OBDA) [1, 2] is concerned with providing end-users and applications with a way to query legacy databases through a high-level ontology that models both the business logic and the underlying data sources. Modern knowledge-based applications have replaced the representation of business logic by using a high-level representation of the business intelligence which is decoupled from the application code. This allows for improved flexibility. In Semantic Web applications [3], the business intelligence is represented by ontologies expressed in the Web Ontology Language 2 (OWL 2) [4]. Briefly, an ontology is a logical theory formed by a collection of concepts and roles and also a set of concept and role assertions [5]. The relationship holding among the concepts and roles in the ontology are described in terms of inclusion and equality axioms. Ontologies used to represent business logic are then used by ontology reasoners to draw conclusions. The conclusions that can be got include making explicit the implicit terminology of concepts defined by the ontology, determining if a certain individual is a member of a concept, or determining if two individuals are related through a role, determining if a concept is subsumed by other concept, or if a role is subsumed by other role.

Thus, the classic OBDA architecture is composed of a global database, a legacy database and a bridge between the ontology and the database. The bridge between the ontology and the data sources is addressed by mappings that define how to express records of the database as ontological assertions. Relational databases are comprised of relations (tables), that in term are defined by data schemas, which define the names and domains of table attributes as well as any integrity constraints that might apply to them, and are composed of records. Ontologies, on the other hand, are composed of axioms, and concept and roles assertions. The mappings define how to populate the ontology in terms of the elements of the database. Basically, the concept and role fillers are defined by SQL queries that indicate how to populate them. Notice that in the case of having several databases, a federation system can be used that allows to see the set of databases as a unified database. In this work, however, we will not take this possibility into account.

In this research, we are concerned with providing with tools for performing OBDA with relational and non-relational data sources. Several tools have been developed by other research groups (see for instance [6–9] that we reviewed in [10]). Some of those tools are closed-source while others are open-source, some are downloadable and can be used as stand-alone applications or as programming libraries. While many times they are a good starting point for building applications, many times they are not flexible enough. In that regard, we are developing a tool, which nowadays is in a prototypical state, that is able to access an H2 database, allowing the user to explicitly formulate mappings, and populating an ontology that can be saved for later querying and visualization. See [11, 10] for previous reports on the functionality of the application and its prospective application areas.

In this paper, we report on the advances we have made on the development of such a tool. In particular, we have added a form that allows end-users to fully specify in a high-level manner the nature of mappings and by writing SQL queries as well. We also added a module that allows to test how our application behaves in the presence of increasing demands. We assume that the reader has a basic knowledge of Description Logics (DL) [12], relational databases [13] and the Web Ontology Language [4].

The rest of the paper is structured as follows. In Sect. 2, we briefly recapitulate the concepts associated with materializing ontologies from tables. In Sect. 3, we present a novel development in the system that allows a naïve user to define a mapping from tables to ontologies in a visual manner. In Sect. 4, we show an empirical evaluation of the performance of the prototype creating tables and ontologies. In Sect. 5, we review related work. In Sect. 6, we conclude and foresee future work.

## 2 Materialization of OWL Ontologies from Relational Databases

Materializing an OWL ontology from a relational database requires exporting the database contents as a text file in OWL format. For doing this, we need to export the schema information of each table as Tbox axioms and the instance data of the tables as Abox assertions. Here, we review the formalization for exporting database relations as ontologies as we presented it in [10] according to the directions given by [1, 14]. Building an ontology from a database requires creating at least a class $C_T$ for every table $T$, and for every attribute $a$ of domain $d$ in $T$ we need two inclusion DL axioms $C_T \sqsubseteq \exists a$ and $\exists a^- \sqsubseteq d$. Primary key values $k_i$ serve the purpose of establishing the membership of individuals to classes as DL Abox asssertions of the form $C_T(C_T\#k^j)$. For indicating that $a^j$ is the value of attribute $a$, we will use a role expression of the form $C_T\#a(C_T\#k^j, C_T\#a^j)$. When it is clear from context, we might drop the prefix $C_T\#$ for simplifying our notation. A foreign key $fk$ in table $T_1$ referencing a primary key field in table $T_2$ will also require to add two Tbox axioms $C_{T_1} \sqsubseteq \exists \mathsf{ref\_}\mathit{fk}$ and $\exists \mathsf{ref\_}\mathit{fk}^- \sqsubseteq C_{T_2}$ and an Abox assertion $\mathsf{ref\_}\mathit{fk}(k^j, fk^t)$ for expressing that the individual named $k^j$ in $C_{T_1}$ is related to the individual named $fk^t$ in $C_{T_2}$. Besides, in any case, if we want to consider a subset of a table for its mapping into an ontology, we might define an SQL query that will act as an SQL filter. In this work, we will only deal with the translation of single tables into OWL, as defined next (for details see [10]):

**Definition 1 (Mapping of a table with a single primary key).** *Let $T$ be a table with schema $T(\underline{k}, a_1, \dots, a_n)$ and instance $\{(k^1, a_1^1, \dots, a_n^1), \dots, (k^m, a_1^m, \dots, a_n^m)\}$. To map $T$ into a DL terminology $\mathcal{T}$, we have to create a class $T$ and for each attribute $a_i$ of domain $D_i$ we have to add two axioms: $T \sqsubseteq \exists a_i$, indicating that every $T$ has an attribute $a_i$, and $\exists a_i^- \sqsubseteq D_i$, meaning that the domain of $a_i$ is $D_i$. The assertional box $\mathcal{A}$ for $T$ will contain $\{T(k^1), \dots, T(k^m)\}$. Given a key value $k_j$, $j = 1, \dots, m$, for every attribute $a_i$, $i = 1, \dots, n$, of the schema and instance value $a_i^j$ (i.e. the value of i-th attribute of the j-th individual), produce a property $a_i(k^j, a_i^j)$.*

*Example 1.* Consider a table for representing people with schema

$$\mathsf{Person}(\underline{\mathsf{personID}}, \mathsf{name}, \mathsf{sex}, \mathsf{birthDate}, \mathsf{weight})$$

and instance as in Fig. 1. This table is created by the SQL script presented in Fig. 2.

| *personID* | *name* | *sex* | *birthDate* | *weight* |
|---|---|---|---|---|
| 1 | John | true | 2010-01-01 | 100.0 |
| 2 | Mary | false | 2009-01-01 | 60.0 |

**Fig. 1.** Relational instance of the table *Person*

```
create table "Person" (
"personID" int unsigned not null auto_increment primary key,
"name" varchar(20) not null,
"sex" boolean,
"birthDate" date,
"weight" real,
);

insert into "Person"("name", "sex", "birthDate", "weight") values ('John', true, '2010-01-01', 100.0);
insert into "Person"("name", "sex", "birthDate", "weight") values ('Mary', false, '2009-01-01', 60.0);
```

**Fig. 2.** SQL script for creating the table Person

The table Person is interpreted in Description Logics according to Def. 1, as $\Sigma = (\mathcal{T}, \mathcal{A})$ where:

$$\mathcal{T} = \left\{ \begin{array}{ll} \mathsf{Person} \sqsubseteq \exists \mathsf{personID}, & \exists \mathsf{personID}^- \sqsubseteq \mathsf{Integer}, \\ \mathsf{Person} \sqsubseteq \exists \mathsf{name}, & \exists \mathsf{name}^- \sqsubseteq \mathsf{String}, \\ \mathsf{Person} \sqsubseteq \exists \mathsf{sex}, & \exists \mathsf{sex}^- \sqsubseteq \mathsf{Boolean}, \\ \mathsf{Person} \sqsubseteq \exists \mathsf{birthDate}, & \exists \mathsf{birthDate}^- \sqsubseteq \mathsf{Date}, \\ \mathsf{Person} \sqsubseteq \exists \mathsf{weight}, & \exists \mathsf{weight}^- \sqsubseteq \mathsf{Real} \end{array} \right\} \text{ and}$$

$$\mathcal{A} = \left\{ \begin{array}{ll} \mathsf{Person}(\mathsf{Person\#1}), & \mathsf{personID}(\mathsf{Person\#1}, 1), \\ \mathsf{name}(\mathsf{Person\#1}, \mathsf{John}), & \mathsf{sex}(\mathsf{Person\#1}, \mathsf{true}), \\ \mathsf{birthDate}(\mathsf{Person\#1}, 2001\text{-}01\text{-}01), & \mathsf{weight}(\mathsf{Person\#1}, 100.0), \\ \mathsf{Person}(\mathsf{Person\#2}), & \mathsf{personID}(\mathsf{Person\#2}, 2), \\ \mathsf{name}(\mathsf{Person\#2}, \mathsf{Mary}), & \mathsf{sex}(\mathsf{Person\#2}, \mathsf{false}), \\ \mathsf{birthDate}(\mathsf{Person\#2}, 2009\text{-}01\text{-}01), & \mathsf{weight}(\mathsf{Person\#2}, 60.0) \end{array} \right\}.$$

Description Logic ontologies are implemented in the OWL language, which includes an XML serialization which we partially present in Fig. 3 by showing the representation for John.

## 3 Visual Mapping Specification

The specification of the mappings for obtaining a the fillers of concept from a table is usually a complex matter for naïve end-users. Remember that a mapping is basically a SQL query that defines how the fillers of concept, property or role are computed in terms of the contents of a database. When there is no support for composing mappings, the user has to write such SQL from scratch. We believe that adding support for building the mappings will improve the user experience of a prospective user of OBDA technology.

With the idea of providing support to end-users in their quest of creating concepts for populating ontologies from database contents, we created a module

```
<owl:Class rdf:about="http://cs.uns.edu.ar/~sag#Person"/>
<!-- http://cs.uns.edu.ar/~sag/Person/personid=1 -->

<owl:NamedIndividual rdf:about="http://cs.uns.edu.ar/~sag/Person/personid=1">
    <rdf:type rdf:resource="http://cs.uns.edu.ar/~sag#Person"/>
    <Person:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
            2010-01-01T00:00:00</Person:birthDate>
    <Person:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">John</Person:name>
    <Person:personID rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</Person:personID>
    <Person:sex rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</Person:sex>
    <Person:weight rdf:datatype="http://www.w3.org/2001/XMLSchema#double">100.0</Person:weight>
</owl:NamedIndividual>
```

**Fig. 3.** Part of the OWL code for the definition of the class Person

that allows to visually specify a mapping from a table. The module retrieves the tables from the database, and allows to select a table. Once the table is selected, its fields can be selected too. The user can then introduce what conditions each field of the table has to satisfy. Besides, one field (usually the key field of the table) has to be selected to fill the concept. The module then will automatically generate the SQL filter for filling the concept by extracting the records from the table, and will also add a subclass axiom to the ontology.

*Example 2.* Consider again the table Person from Ex. 1 and suppose that some user of the system wants to define the concept "heavy, young, male individual". Suppose also that the user models a heavy individual as somebody who weighs at least a hundred kilograms, a young individual as someone who was born after 2001, and a male individual as someone of male sex. People of male sex are codified as having the column named sex as true while females are codified as false. Although this is a trivial example, it shows the complexities that run into database modeling that produce a degradation of the representation of the world and that are unretrievable afterwards. The user will then visually specify the conditions for an individual to be a member of the concept YoungHeavyMalePerson in a form like the one presented in Fig. 4. Notice how the user specifies which database field corresponds to the key (i.e. the name of the individuals), in this case personID. In turn, the system will generate a SQL query as follows:

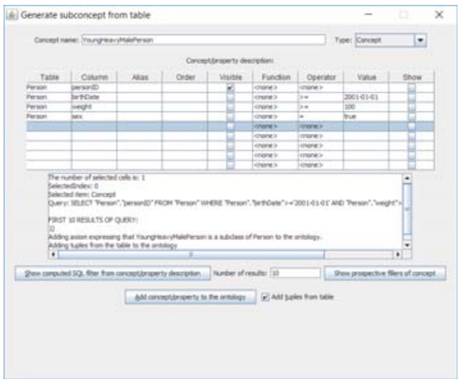
```
SELECT "Person"."personID" FROM "Person"
WHERE "Person"."birthDate" >= '2001-01-01'
AND "Person"."weight" >= 100 AND "Person"."sex" = true
```

After the execution of the query that will compute the individuals that fill the concept, the system will add to the current ontology the triples expressing that those individuals are the fillers of the concept YoungHeavyMalePerson. Besides, in order to relate this concept to its superconcept, the axiom

$$\mathsf{YoungHeavyMalePerson} \sqsubseteq \mathsf{Person}$$

will be added to the current ontology as well.

This will lead to the situation presented in Fig. 5. The new class YoungHeavyMalePerson is defined as a subclass of Person and John, whose personID role is "1" becomes


Generate subconcept from table
Show computed SQL filter from concept/property description
Show prospective fillers of concept
Add concept/property to the ontology

Concept name: YoungHeavyMalePerson Type: Concept

Concept/property description:

| Table | Column | Alias | Order | Visible | Function | Operator | Value | Show |
|---|---|---|---|---|---|---|---|---|
| Person | personID | | | ✓ | <none> | <none> | | |
| Person | birthDate | | | | <none> | >= | 2001-01-01 | |
| Person | weight | | | | <none> | >= | 100 | |
| Person | sex | | | | <none> | = | true | |
| | | | | | <none> | <none> | | |
| | | | | | <none> | <none> | | |
| | | | | | <none> | <none> | | |
| | | | | | <none> | <none> | | |
| | | | | | <none> | <none> | | |

```
The number of selected cells is: 1
SelectedIndex: 0
Selected item: Concept
Query: SELECT "Person"."personID" FROM "Person" WHERE "Person"."birthDate">='2001-01-01' AND "Person"."weight">

FIRST 10 RESULTS OF QUERY:
[1]
Adding axiom expressing that YoungHeavyMalePerson is a subclass of Person to the ontology.
Adding tuples from the table to the ontology
```

Number of results: 10

[✓] Add tuples from table

**Fig. 4.** Visual concept specification of the concept YoungHeavyMalePerson

a member of YoungHeavyMalePerson. Notice also that no new individuals are defined as John is already present in the ontology because he is a Person. In this sense, we adhere to the unique name assumption as much as we can although this is not required by the formalism. Also notice how the intensional definition of the concept is lost in the ontology (other than being a subclass of Person) and only its extension is maintained in the ontology (as the set of its fillers).

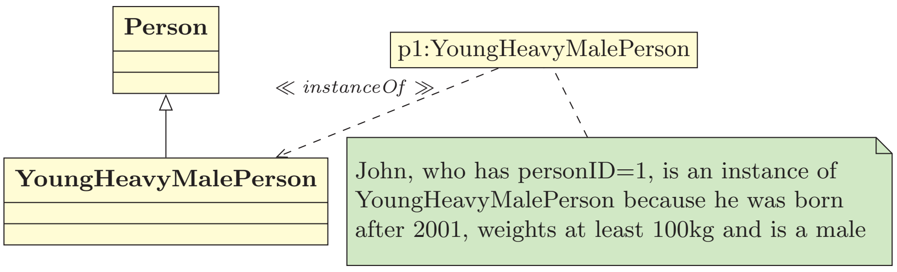

Person
p1:YoungHeavyMalePerson
≪ instanceOf ≫
YoungHeavyMalePerson
John, who has personID=1, is an instance of YoungHeavyMalePerson because he was born after 2001, weights at least 100kg and is a male

**Fig. 5.** Situation arisen by specifying a subclass of Person named YoungHeavyMalePerson

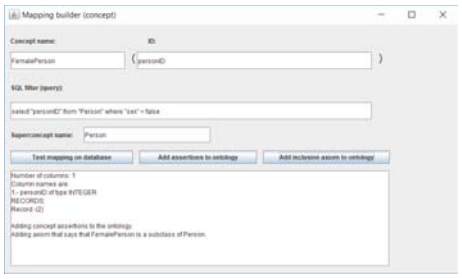
Another feature that the current version of the system includes is the possibility of specifying a subclass by means of an explicit SQL query.

*Example 3.* Continuing Ex. 2, the concept FemalePerson (which defines a subset of the table Person formed by women) is specified by means of the SQL query:

```
select "personID" from "Person" where "sex" = false
```

This can be done by using the form presented in Fig. 6. Notice the additional OWL code in the ontology generated by out tool which is presented in Fig. 7

expressing that a female person is a person (i.e. $\mathsf{FemalePerson} \sqsubseteq \mathsf{Person}$ is an axiom in the ontology) and that Mary is both a female person and a person (i.e. $\mathsf{FemalePerson}(\mathsf{Mary})$ and $\mathsf{Person}(\mathsf{Mary})$ are assertions in the ontology).


Mapping builder (concept)
Concept name:
ID:
SQL filter (query):
Superconcept name:
Test mapping on database
Add assertions to ontology
Add inclusion axiom to ontology

Concept name: FemalePerson ( ID: personID )

SQL filter (query): select "personID" from "Person" where "sex" = false

Superconcept name: Person

```
Number of columns: 1
Column names are:
1 - personID of type INTEGER
RECORDS:
Record: (2)

Adding concept assertions to the ontology
Adding axiom that says that FemalePerson is a subclass of Person.
```

**Fig. 6.** Specification of the subclass FemalePerson of Person by a SQL query

```
<owl:Class rdf:about="http://cs.uns.edu.ar/~sag#FemalePerson">
    <rdfs:subClassOf rdf:resource="http://cs.uns.edu.ar/~sag#Person"/>
</owl:Class>
...
<owl:NamedIndividual rdf:about="http://cs.uns.edu.ar/~sag/FemalePerson=2">
    <rdf:type rdf:resource="http://cs.uns.edu.ar/~sag#FemalePerson"/>
...
</owl:NamedIndividual>
```

**Fig. 7.** Portion of OWL code for introducing subconcept FemalePerson

## 4 Experimental Evaluation

We now discuss some of the tests we have performed in order to test how our application handles increasing demands in database size. The performance of our system is affected mainly by the fact that we chose to materialize tables as triples (i.e. class membership, property and roles assertions) and also by three factors: (i) the system is implemented in the JAVA programming language; (ii) the database management system that we use is H2[1], and, (iii) the handling of the global ontology is done via the OWL API [15, 16].

Our tests were conducted on an ASUS notebook having an Intel Core i7, 3.5GHz CPU, 8GB RAM, 1TB HDD, Windows 10. They involved the creation of simple databases composed by a single table containing 100 fields of text type filled with an increasing number of records. Table 1 summarizes our results.

[1] See `http://www.h2database.com`.

**Table 1.** Running times for ontology generation

| *Number of records* | *Database file size in Megabytes* | *Time for creating the ontology in seconds* | *Size of the ontology in Megabytes* | *Time for loading the ontology from disk in seconds* |
|---|---|---|---|---|
| 1,000 | 0.80 | 1.029 | 8.65 | 4.014 |
| 10,000 | 8.82 | 5.345 | 87.26 | 15.106 |
| 100,000 | 98.11 | 66.48 | 19,053.36 | out of memory error |
| 1,000,000 | 1,080.60 | out of memory error | - | - |

As it can be seen, our implementation starts having problems at tables with 100,000 records; although an ontology can be generated and saved to the disk, when we try to load the ontology we saved previously, we get an error inside the code of the OWL API, indicating that that library cannot handle such a data load. When running a test for creating a database of a million records, the H2 database produces an error (which is understandable as it is maintained in RAM). Therefore, we conclude that our application can only handle tables with a size tens of thousands records and is not able of handling tables of a hundred thousand records. Because of this limitation, we think that we will be forced to use query-rewriting techniques [1] for delegating the evaluation of queries to the database management system instead of the ontology management library.

## 5 Related Work

With ViziQuer, Cerans et al. [17] provide an open source tool for web-based creation and execution of visual diagrammatic queries over RDF/SPARQL data. The tool supports the data instance level and statistics queries, providing visual counterparts for most of SPARQL 1.1 select query constructs, including aggregation and subqueries. A query environment can be created over a user-supplied SPARQL endpoint with known data schema. ViziQuer provides a visual interface for expressing user queries in SPARQL posed against an ontology. In contrast, we provide the user with an interface for describing subclass expressions and inclusion axioms by means of restrictions imposed on records of a relational table with the aim of populating an ontology that could later be exposed and queried as an SPARQL endpoint.

Christodoulou et al. [18] make the case that structural summaries over linked-data sources can inform query formulation and provide support for data integration and query processing over multiple linked-data sources. To fulfil this aim, they propose an approach that builds on a hierarchical clustering algorithm for inferring structural summaries over linked-data sources. Thus, their approach takes as input an RDF repository and then reverse engineers an ontology using clustering techniques to detect prospective classes. In contrast, we take a database and the user proposes SQL queries to express subconcepts intensionally; when the SQL queries are executed, the fillers of the concept populate the ontology building an extensional de facto definition of the concept. In that regard, the work of Christodoulou et al. can be considered complementary to ours.

Barrasa et al. [19] propose R2O, an extensible and declarative language to describe mappings between relational DB schemas and ontologies implemented in RDF(S) or OWL. R2O provides an extensible set of primitives with well defined semantics. This language has been conceived expressive enough to cope with complex mapping cases arisen from situations of low similarity between the ontology and the DB models. R2O allows the user to express complex queries in terms of ontologies in a language that is similar to the relational algebra but aimed at ontologies. Therefore, this approach is complementary to ours because it allows to query the ontology once it is published in OWL format.

## 6 Conclusions and Future Work

We presented the advances we have made in our framework for performing ontology-based data access by means of performing a materialization approach. Our implementation is JAVA-based and relies on a H2 database management system and a JAVA library called OWL-API for accessing and querying databases and maintaining an OWL database in main memory, respectively. We presented several enhancements that we have made to the previous iteration of our prototype implementation, which now includes a visual mapping specification functionality and allows to maintain a global database that can be either created from scratch or loaded from disk, modified and later saved again to disk. From the experimental evaluation to which we subjected our system, we conclude that our application is able to handle a moderate workload of a table of tens of thousands of records but fails to handle table of the order of millions of records. In this regard, we think that we will be forced to use query-rewriting techniques for delegating the evaluation of queries to the database management system instead of the ontology management library. Part of our current research efforts are aimed in this direction. Other form of improvement lies in the possibility of addressing the federation of databases for performing integration of multiple heterogeneous data sources.

**Acknowledgments.** This research is funded by Secretaría General de Ciencia y Técnica, Universidad Nacional del Sur, Argentina and by Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC-PBA).

## References

1. Kontchakov, R., Rodríguez-Muro, M., Zakharyaschev, M.: Ontology-Based Data Access with Databases: A Short Course. In: Reasoning Web: Semantic Technologies for Intelligent Data Access of the LNCS. Volume 8067. Springer (2013) 194–229
2. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-Based Data Access – A Survey. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18). (2018) 5511–5519

3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
4. Bao, J., Kendall, E.F., McGuinness, D.L., Patel-Schneider, P.F.: OWL 2 Web Ontology Language Quick Reference Guide (Second Edition) W3C Recommendation 11 December 2012 (2012)
5. Gruber, T.R.: A translation approach to portable ontologies. Knowledge Acquisition **5**(2) (1993) 199–220
6. Calvanese, D., Giacomo, G.D., Lembo, D., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web **2**(1) (2011) 43–53
7. Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: BootOX: Practical Mapping of RDBs to OWL 2. In: the 14th International Semantic Web Conference. (2015) 113–132
8. de Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: Automatic R2RML mapping generation from relational databases (2015)
9. Pinkel, C., Binnig, C., Jiménez-Ruiz, E., Kharlamov, E., May, W., Nikolov, A., Bastinos, A.S., Skjæveland, M.G., Solimando, A., Taheriyan, M., Heupel, C., Horrocks, I.: RODI: Benchmarking Relational-to-Ontology Mapping Generation Quality. Semantic Web (2016) 1–26
10. Gómez, S.A., Fillottrani, P.R.: Towards a Framework for Ontology-Based Data Access: Materialization of OWL Ontologies from Relational Databases. In Pesado, P., Aciti, C., eds.: X Workshop en Innovación en Sistemas de Software (WISS 2018), XXIV Congreso Argentino de Ciencias de la Computación CACIC 2018. (2018) 857–866
11. Gómez, S.A., Fillottrani, P., Estévez, E., Pesado, P., Pasini, A., Muñoz, R., Thomas, P.: Desarrollo de herramientas para acceso a bases de datos heterogéneas basado en ontologías en el contexto de la entrega de servicios públicos digitales. Primer Encuentro de Centros Propios y Asociados de la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (2018) 235–238
12. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook – Theory, Implementation and Applications. Cambridge University Press (2003)
13. Silberschatz, A., Korth, H.F., , Sudarshan, S.: Database System Concepts (6th edition). McGraw-Hill Education (1983)
14. Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J.: A Direct Mapping of Relational Data to RDF. W3C Recommendation 27 September 2012 (2012)
15. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL Ontologies. Semantic Web **2**(1) (2011) 11–21
16. Matentzoglu, N., Palmisano, I.: An Introduction to the OWL API. Technical report, The University of Manchester (2016)
17. Cerans, K., Sostaks, A., Bojars, U., Ovcinnikova, J., Lace, L., Grasmanis, M., Romane, A., Sprogis, A., Barzdiņs, J.: ViziQuer: A Web-Based Tool for Visual Diagrammatic Queries Over RDF Data. In et al., G.A., ed.: The Semantic Web: ESWC 2018 Satellite Events. ESWC 2018. Lecture Notes in Computer Science. Volume 11155., Springer, Cham (2018) 158–163
18. Christodoulou, K., Paton, N.W., Fernandes, A.A.: Structure Inference for Linked Data Sources using Clustering. In: EDBT '13 Proceedings of the Joint EDBT/ICDT 2013 Workshops. (2013) 60–67
19. Barrasa, J., Corcho, Ó., Gómez-Pérez, A.: R2O, an extensible and semantically based database-to-ontology mapping language. Proceedings of the Second Workshop on Semantic Web and Databases, SWDB 2004 **3372** (2004)