A Reference Architecture for Building Semantic-Web Mediators*

Carlos R. Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo

University of Sevilla, Spain
{carlosrivero,inmahernandez,druiz,corchu}@us.es

Abstract. The Semantic Web comprises a large amount of distributed and heterogeneous ontologies, which have been developed by different communities, and there exists a need to integrate them. Mediators are pieces of software that help to perform this integration, which have been widely studied in the context of nested relational models. Unfortunately, mediators for databases that are modelled using ontologies have not been so widely studied. In this paper, we present a reference architecture for building semantic-web mediators. To the best of our knowledge, this is the first reference architecture in the bibliography that solves the integration problem as a whole, contrarily to existing approaches that focus on specific problems. Furthermore, we describe a case study that is contextualised in the digital libraries domain in which we realise the benefits of our reference architecture. Finally, we identify a number of best practices to build semantic-web mediators.

Keywords: Information Integration, Mediator, Semantic-web Technologies.

1 Introduction

The Semantic Web is gaining popularity day by day [48]. The best proof of this popularity is that there are a number of domains in which semantic-web technologies are becoming a de facto standard for representing and exchanging data, e.g., the Gene Ontology in the life sciences domain [4], SwetoDBLP in the digital libraries domain [1], FOAF in the people description domain [12], or DBPedia in multiple domains [6].

Semantic-web technologies comprise RDF, RDFS and OWL ontology languages to represent models and the data that populates them, and the SPARQL query language to query these data [3]. Ontologies are shared models by a number of communities that have been developed by consensus [8]. The development of an ontology is not a trivial task: to reach an agreement amongst one or more communities can be an unaffordable problem [22]. Due to this fact, there are a

* Supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

C. Salinesi and O. Pastor (Eds.): CAiSE 2011 Workshops, LNBIP 83, pp. 330–341, 2011.
© Springer-Verlag Berlin Heidelberg 2011

large variety of heterogeneous and distributed ontologies in the Semantic Web, and there is a need to integrate them [20,34].

Mediators are pieces of software that help to integrate data models [27]. This integration comprises two tasks: data integration and data translation. The former deals with answering queries posed over a target model using a set of source models only [13,21,27,52]. In this task, the target model is virtual and contains no data. The latter, also known as data exchange, consists of, based on user preferences, populating a target model with data that comes from a set of source models [15,18,36,47]. Therefore, the aim of this task is the materialisation of the target model. To perform these tasks, mediators rely on the use of mappings, which are relationships between source and target models [18,27,39].

Building and maintaining mappings is costly since users must write them, check whether they work as expected or not, making changes if necessary, and restart the loop [5,37]. Therefore, it is appealing to provide techniques for building and maintaining mappings automatically. To solve this problem, there are a number of approaches in the bibliography to automatically generate correspondences, i.e., uninterpreted mappings that have to be interpreted to perform the integration between source and target models [17,43].

Furthermore, other approaches are based on automatically-generated executable mappings, which encode the interpretation of correspondences into a query in a given query language, such as SQL, XSLT, XQuery or SPARQL [31,38,39,40]. Therefore, the data integration task is performed by rewriting queries over these executable mappings, and the data translation task is performed by executing them by means of a query engine.

Mediators have been widely studied in the context of (nested) relational models which represent trees [18,21,27,39,52]. Unfortunately, these mediators are not applicable to the semantic-web context due to a number of inherent differences between databases and ontologies [25,32,33,51]. Some examples of these inherent differences are the following:

- Structure: a nested relational model represents a tree in which there is a unique path from the root to every node. Contrarily, an ontology forms a graph in which may be zero, one or more paths connecting every two nodes.
- Optionality: in a nested relational model, an instance of a nested node exists if and only if the entire path from this node to the root exists. Contrarily, in an ontology, elements are optional by default, so it is not possible to assume that there exists a path that connects every two nodes.

In this paper, we present a reference architecture for building semantic-web mediators to solve these inherent differences. To the best of our knowledge, this is the first reference architecture in the bibliography that solves the integration problem as a whole, contrarily to existing approaches that focus on specific problems [15,20,28,29,30,47]. Furthermore, we survey a number of approaches in the bibliography that can be used to build various modules within our architecture. We also present a case study that is contextualised in the domain of digital libraries. Finally, after analysing existing approaches in the bibliography

and devising our reference architecture, we identify a number of best practices to build semantic-web mediators.

This paper is organised as follows: Section 2 describes the related work. Section 3 presents our reference architecture for building semantic-web mediators. In Section 4, we describe a case study that is contextualised in the digital libraries domain. Finally, Section 5 recaps our main conclusions and present a number of best practices to build semantic-web mediators.

2 Related Work

In this section, we survey a number of approaches in the bibliography for building semantic-web mediators.

Maedche et al. [28,29] proposed a mediator that is based on the Semantic Bridge Ontology, i.e., an ontology that bridges the gap between source and target ontologies. This ontology allows to relate classes, properties, and instances by means of semantic bridges. Furthermore, they allow to combine these semantic bridges by means of specialisation, abstraction, composition or alternative. This mediator performs the data translation task using an ad-hoc technique, which consists of evaluating the instances of the Semantic Bridge Ontology to translate the data of the source ontologies into the target.

The mediator proposed by Dou et al. [15] is based on executable mappings, which are specified by hand and represented with the Web-PDDL language (that is a subset of first-order logic) devised by the authors. The mediator performs the data translation task by merging source and target ontologies into a single global ontology. Then, a reasoner devised by the authors is used over this global ontology, and it retains conclusions that are only expressed in the vocabulary of the target ontology. These conclusions compound the resulting target data.

Haase and Wang [20] proposed a mediator that is based on a reasoner devised by the authors, which is able to deal with distributed ontologies. Their approach builds on executable mappings that are expressed in OWL-DL, and it is assumed that they are known beforehand. A virtual ontology is created that merges source and target ontologies and the executable mappings. Finally, the data integration task is performed by reasoning over this virtual ontology with the user query, retrieving the target query results.

The mediator proposed by Serafini and Tamilin [47] focuses on the translation of class instances. Therefore, they use two types of correspondences, between classes and between individuals, and both are expressed using OWL-DL. In this approach, the data translation task is performed by using a reasoner to reclassify classes or individuals from the source ontology into the target.

Makris et al. [30] devised a mediator based on OWL that performs the data integration task by means of handcrafted executable mappings. These mappings are expressed using the alignment API proposed by Euzenat [16]. Then, a SPARQL query over the target ontology is rewritten using an ad-hoc query rewriting technique, which deals with graph pattern operators such as AND, UNION or OPTIONAL, and even with FILTER expressions.

To conclude, none of the existing approaches, as far as we know, solve the integration problem as a whole. They focus on algorithms to solve the data integration and/or data translation tasks, but none of them presents an architecture that includes all the problems that must be solved when building a mediator: the generation of mappings, the data integration task, the data translation task, and the technology to support it.

3 Reference Architecture

In this section, we describe our reference architecture to build semantic-web mediators. Furthermore, we identify a number of approaches in the bibliography that can be used to build various modules of this architecture.

Figure 1 shows our reference architecture that is divided in four main modules: Setup, Data Integration, Data Translation and Support. The Setup module deals with the automatic generation of mappings that relate source and target ontologies. Data Integration is responsible for retrieving data from the source ontologies by means of a query over the target ontology. The Data Translation module takes the data from the source ontologies and compounds it to generate target data. Finally, the Support module comprises a number of auxiliary components that are needed in the integration process. In the following subsections we describe these modules.

3.1 Setup

This module takes the source and target ontologies as input, and produces a set of mappings based on these ontologies.

Correspondence Generation deals with the process of specifying correspondences that may be defined in different ways, namely: handcrafted [15,30], handcrafted with the help of a graphical tool [2,42], automatically-generated [17,43], or automatically-generated by means of patterns [44].

Mapping Generation takes the source and target ontologies and a set of correspondences as input, and generate a number of mappings as output. Some approaches use correspondences as mappings, which must be interpreted to perform the data integration and data translation tasks. Existing semantic-web mediators interpret correspondences by means of ad-hoc techniques or reasoners, which hinder their applicability to real-world scenarios due to the low performance with medium or large amount of data of ad-hoc techniques and reasoners [15,20].

Other approaches use executable mappings that encode the interpretation of correspondences. The semantic-web mediator devised by Dou et al. [15] is based on handcrafted executable mappings. Popa et al. [39] devised a technique to generate executable mappings for nested relational models. In this technique, correspondences that are related by source and/or target restrictions are grouped to form executable mappings. Unfortunately, it is not applicable to the semantic-web context due to the inherent differences previously described. Qin et al. [40] devised a technique to generate executable mappings between OWL ontologies,

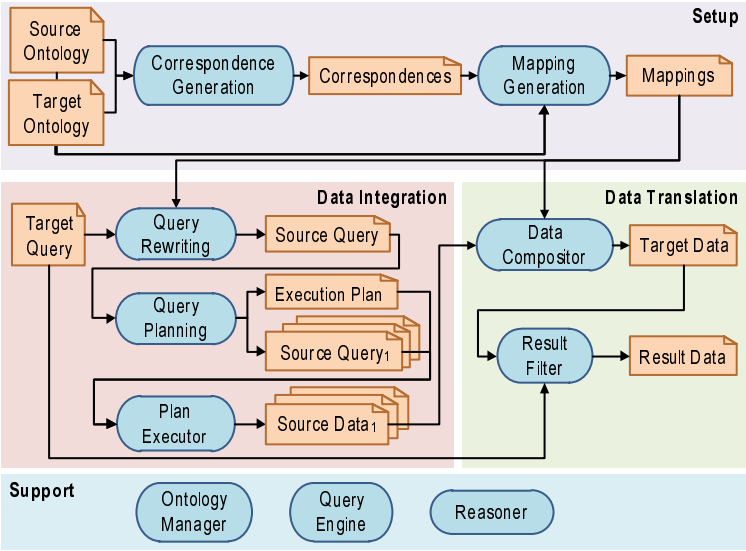

Fig. 1. Reference architecture of a semantic-web mediator based on executable mappings

which are represented in Datalog or SWRL. This technique takes a set of correspondences and a set of instance examples of the target ontology as input.

Polleres et al. [38] proposed the use of the SPARQL query language to represent executable mappings, but it has a number of issues, such as the optionality of properties, or the multi-typed instances. However, note that current semantic-web technologies have proved to be efficient in the management of large data volumes using SPARQL queries [7,46]. Consequently, technology is not an obstacle to the development of SPARQL executable mappings.

3.2 Data Integration

This module takes the mappings and a query over the target ontology as input, and it is responsible for rewriting and planning this query to retrieve source data. This process is divided into three tasks: Query Rewriting, Query Planning and Plan Executor.

Query Rewriting deals with the reformulation of the target query into a single query over source ontologies. This reformulation varies depending on the type of mappings. GAV mappings comprise one single clause of the target ontology and a number of clauses of the source ontologies [27]. In this case, the reformulation is performed by unfolding these mappings into the user query [35]. LAV mappings comprise a number of clauses of the target ontology and one single clause of one source ontology [27]. In this case, the reformulation is performed by applying the techniques of answering queries using views [21]. Finally, GLAV mappings comprise a number of clauses of both source and target ontologies [19]. In this case,

the reformulation is performed using hybrid techniques from GAV and LAV [52]. Note that these techniques focus on nested relational models (specifically, Datalog and XML [21,52]); however, there is an increasing interest on SPARQL query rewriting in the semantic-web community [13,24].

Query Planning divides the source query into a number of queries, each of which affects only one source ontology. Furthermore, it generates a plan that specifies how these queries must be executed. In this context, Ives et al. [23] proposed a query planner that exploits the features of the source data for XML models. Thakkar et al. [50] focused on the automatic generation of parameterised executions plans that are exported as web services. Furthermore, they devised techniques that improve the efficiency of these plans by reducing the number of requests to data sources. Braga et al. [9] presented a framework to answer multi-domain queries, which can be answered by combining the data of one or more sources. Finally, Langegger et al. [26] and Quilitz and Leser [41] proposed techniques to answer SPARQL queries over distributed RDF sources.

Plan Executor is responsible for executing the previous plans by posing queries over source ontologies and retrieving their data. In this case, if the sources are web-based, it is mandatory to use wrappers to access to them [14]. These wrappers deal with one or more of the following problems: 1) form filling, in which the query over the source is used to fill in the search forms of the web application appropriately; 2) navigation, in which the resulting page after submitting a search form is navigated until pages that contain the information of interest are found; 3) information extraction, in which the information of interest is extracted and structured according to a given ontology; 4) verification, in which the retrieved information is checked for errors.

3.3 Data Translation

This module takes the mappings, the retrieved source data and the target query as input, and it is responsible for composing the result data of the target query.

Data Compositor takes the mappings and the source data as input, and generates target data as output. When using executable mappings, this task consists of executing these mappings by means of a query engine over source data to produce target data [39]. When using correspondences, this task is performed by interpreting correspondences using ad-hoc techniques or reasoners [15,28]. Note also that executable mappings can be automatically optimised by the query engine to achieve a better performance [31]. Furthermore, executable mappings and the query engine are independent, so we may choose any query engine to perform this task without changing the mappings [31].

Result Filter is needed in some cases since the source data retrieved by Data Integration module may comprise more information than the requested by the user query. Therefore, this task consists of executing the target query over the composed target data and applying other optimisations to this data to produce the final result of the target query.

3.4 Support

This module is orthogonal to the architecture and it provides a number of semantic-web technologies to support the development of semantic-web mediators.

Ontology Manager is responsible for the management of ontologies, which comprises an internal model to represent ontologies, a storage mechanism for ontologies, and operations to load from the file system to this internal model and conversely. Currently, there are a number of frameworks that manage ontologies such as Jena or Sesame [10,11]. Furthermore, it is important to notice that these frameworks have proved to be mature enough to cope with large volumes of data [7,46].

Query Engine deals with the management and execution of queries. Jena and Sesame frameworks incorporate modules to work with SPARQL queries and, consequently, Ontology Manager and Query Engine are implemented using the same semantic-web technologies. In this context, Schmidt et al. [45] studied the theoretical foundations of optimising SPARQL queries, which are the base of these engines.

Finally, Reasoner is able to make the knowledge explicit over a specific ontology, i.e., to infer new triples according to a set of rules. This is mandatory when working with the SPARQL query language since it only deals with plain RDF and does not implement RDFS/OWL semantics [3]. Pellet is an OWL-DL reasoner that is distributed alone [49], but there are other platforms that incorporate reasoners such as Jena or Oracle amongst others.

4 Case Study

In this section, we present a case study that uses a semantic-web mediator for integrating ontologies, which is contextualised in the digital libraries domain.

Figure 2 shows a scenario in which we integrate two source and target ontologies. On the one hand, the source ontology comprises two classes with no relation between them, which are *src:Article* and *src:Paper*. These classes model published articles and papers, respectively. Furthermore, these classes are related to three data properties, namely: *src:title*, which models the title of the article or paper; *src:journal*, which represents the title of the journal in which the article has been published; and *src:conference*, which stands for the conference in which the paper has been published.

On the other hand, the target ontology comprises three classes, which are the following: *tgt:Publication*, *tgt:Article*, which specialises *tgt:Publication*; and *tgt:Paper*, which also specialises *tgt:Publication*. These classes are related to two data properties: *tgt:title* and *tgt:host*. Behind each information integration scenario, there is an intention of change that reflects new user needs, i.e., the source model is the ontology before changes are applied, and the target ontology is the ontology after changes are applied. Behind this case study, there is an intention of change that consists of incorporating the source classes into the hierarchy of the target ontology: the classes in the source ontology are not related and, in the target ontology, these classes are part of a hierarchy.

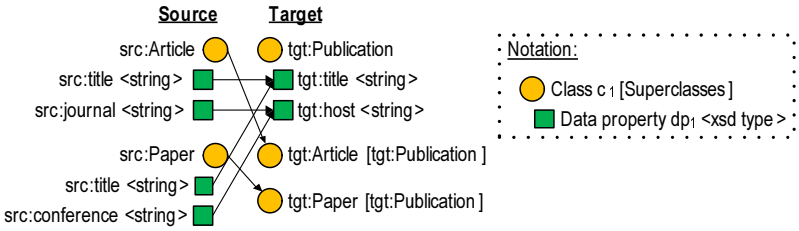

Fig. 2. Example of integration in the digital libraries domain

```
/* Translates Articles and
   titles */
M1:
CONSTRUCT{
  ?tArticle   rdf:type    tgt:Article.
  ?tArticle   tgt:title   ?tTitle.
} WHERE {
  ?sArticle   rdf:type    src:Article.
  ?sArticle   src:title   ?sTitle.
} LET{
  ?tArticle := ?sArticle.
  ?tTitle := ?sTitle.
}
```

```
/* Translates Papers and
   conferences */
M2:
CONSTRUCT{
  ?tPaper   rdf:type        tgt:Paper.
  ?tPaper   tgt:conference  ?tConf.
} WHERE {
  ?sPaper   rdf:type        src:Paper.
  ?sPaper   src:conference  ?sConf.
} LET{
  ?tPaper := ?sPaper.
  ?tConf := ?sConf.
}
```

Fig. 3. SPARQL mappings to integrate the ontologies of the case study

To build a mediator for this case study, the first step consists of building a set of correspondences amongst the elements of source and target ontologies. These correspondences are represented as arrows in Figure 2. Note that these correspondences are only visual aids to help reader to understand the case study. Correspondences are interpreted to get a number of mappings to perform the data integration or data translation tasks. However, the interpretation of these

```
/* Select all publications with
   their titles and hosts */
Q:
SELECT ?title ?host
WHERE {
  ?publication   rdf:type    tgt:Publication.
  ?publication   tgt:title   ?title.
  ?publication   tgt:host    ?host.
}
```

```
Q1:
SELECT ?title ?host
WHERE {
  ?article   rdf:type      src:Article.
  ?article   src:title     ?title.
  ?article   src:journal   ?host.
}
```

```
Q2:
SELECT ?title ?host
WHERE {
  ?paper   rdf:type        src:Paper.
  ?paper   src:title       ?title.
  ?paper   src:conference  ?host.
}
```

Fig. 4. An example of query rewriting in the case study

correspondences is not standardised: "correspondences are interpreted in distinct ways by different [mapping] systems" [39].

Then, the next step consists of automatically generating SPARQL executable mappings based on these correspondences. Figure 3 presents two examples of SPARQL executable mappings. It is important to notice that these mappings are CONSTRUCT queries [3], which comprise three parts: the WHERE, CONSTRUCT and LET clauses. The WHERE clause is used to retrieve data, the CONSTRUCT clause is used to compose data into the form of a RDF graph, and the LET clause contains a number of assignments. Note that these assignments are not allowed in standard SPARQL, but there are some implementations that support them such as the Jena framework.

In this phase, we are able to translate data from the source into the target ontology by executing the SPARQL executable mappings over the source ontologies by means of a query engine. The main benefit of using executable mappings is that the result is the same regardless the order of execution. Therefore, we have to consider no process or workflow to perform this translation: by executing these mappings in any order, we obtain the final target data.

Finally, assume that a user poses a query over the target ontology to retrieve publications with their titles and hosts, as shown in query Q of Figure 4. This query is rewritten and we obtain two queries Q_1 and Q_2, each of which affects one source class only. The rewriting of this query is based on the SPARQL executable mappings previously generated. Note that, in this case, the queries are of the SELECT type, each of which comprises a unique WHERE clause and a set of variables to be returned. This type of query retrieves a plain table of values for a set of triple patterns [3]. In this case, the execution plan is trivial since we pose Q_1 and Q_2, or conversely, to obtain the same result.

5 Conclusions

In this paper, we present a reference architecture to build semantic-web mediators that is, to the best of our knowledge, the first reference architecture in the bibliography that solves the integration problem as a whole, contrarily to existing approaches that focus on specific problems. Our architecture comprises four modules: Setup, Data Integration, Data Translation and Support. Setup module is responsible for generating uninterpreted mappings (also known as correspondences) or executable mappings. Data Integration takes a target query as input, and it deals with dividing the query into a number of queries, which are posed over the sources, and retrieving source data. Data Translation module takes this source data as input, and it interprets correspondences or executes executable mappings over it to produce target data. Finally, Support module deals with orthogonal semantic-web technologies for supporting semantic-web mediators.

After analysing existing approaches in the bibliography and devising our reference architecture, we identify a number of best practices to build semantic-web mediators, which are the following:

Building and maintaining mappings is costly since users must write them, check whether they work as expected or not, making changes if necessary, and

restart the loop [5,37]. Therefore, it is appealing to provide techniques for building and maintaining mappings automatically. To solve this problem, there are a number of approaches in the bibliography to automatically generate correspondences, i.e., uninterpreted mappings that have to be interpreted to perform the integration between source and target models [17,43].

1. Automatic generation of mappings: handcrafted mappings are difficult to build and maintain, since they require the intervention from users. Consequently, semantic-web mediators should use techniques to automatically generate mappings.
2. Exploiting source capabilities in data integration: when rewriting and planning a target query into a number of source queries, semantic-web mediators should restrict as much as possible these source queries to extract as less irrelevant data as possible. Thanks to this, the data translation task is alleviated significantly.
3. Data translation by means of executable mappings: when using executable mappings, the data translation task consists of executing them by means of a query engine over the source data to produce target data. Semantic-web mediators should build on executable mappings since they encode the interpretation of correspondences, and we do not have to rely on ad-hoc and complex techniques to interpret them, i.e., to perform the data integration or data translation tasks.
4. Support for large data volumes: semantic-web data is growing day by day, and to solve real-world integration problems, semantic-web mediators should work with semantic-web technologies that support large data volumes. As we have seen (cf. Section 3), current semantic-web technologies are mature enough to cope with large data volumes.

References

1. Aleman-Meza, B., et al.: SwetoDblp ontology of computer science publications. J. Web Sem. 5(3) (2007)
2. Alexe, B., et al.: Muse: a system for understanding and designing mappings. In: SIGMOD Conference (2008)
3. Antoniou, G., van Harmelen, F.: A Semantic Web Primer, 2nd edn. (2008)
4. Ashburner, M., et al.: Gene Ontology: tool for the unification of biology. Nature genetics 25 (2000)
5. Bernstein, P.A., Haas, L.M.: Information integration in the enterprise. Commun. ACM 51(9) (2008)
6. Bizer, C., et al.: DBpedia - a crystallization point for the web of data. J. Web Sem. (2009)
7. Bizer, C., Schultz, A.: The berlin SPARQL benchmark. In: Int. J. Semantic Web Inf. Syst. (2009)
8. Bouquet, P., et al.: Contextualizing ontologies. J. Web Sem. 1(4) (2004)
9. Braga, D., et al.: Optimization of multi-domain queries on the web. PVLDB 1(1) (2008)

10. Broekstra, J., et al.: Sesame: A generic architecture for storing and querying RDF and RDF schema. In: International Semantic Web Conference (2002)
11. Carroll, J.J., et al.: Jena: implementing the semantic web recommendations. In: WWW (2004)
12. Celma, Ò., Serra, X.: FOAFing the music: Bridging the semantic gap in music recommendation. J. Web Sem. 6(4) (2008)
13. Correndo, G., et al.: SPARQL query rewriting for implementing data integration over linked data. In: EDBT/ICDT Workshops (2010)
14. de Viana, I.F., Hernandez, I., Jiménez, P., Rivero, C.R., Sleiman, H.A.: Integrating deep-web information sources. In: Demazeau, Y., Dignum, F., Corchado, J.M., Bajo, J., Corchuelo, R., Corchado, E., Fernández-Riverola, F., Julián, V.J., Pawlewski, P., Campbell, A. (eds.) Trends in PAAMS. Advances in Intelligent and Soft Computing, vol. 71, pp. 311–320. Springer, Heidelberg (2010)
15. Dou, D., et al.: Ontology translation on the semantic web. J. Data Semantics 2 (2005)
16. Euzenat, J.: An API for ontology alignment. In: International Semantic Web Conference (2004)
17. Euzenat, J., Shvaiko, P.: Ontology matching (2007)
18. Fagin, R., et al.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1) (2005)
19. Friedman, M., et al.: Navigational plans for data integration. In: AAAI (1999)
20. Haase, P., Wang, Y.: A decentralized infrastructure for query answering over distributed ontologies. In: ACM Symposium on Applied Computing (2007)
21. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. 10(4) (2001)
22. Halevy, A.Y., et al.: Piazza: data management infrastructure for semantic web applications. In: WWW (2003)
23. Ives, Z.G., et al.: Adapting to source properties in processing data integration queries. In: SIGMOD Conference (2004)
24. Jing, Y., et al.: SPARQL graph pattern rewriting for OWL-DL inference queries. Knowl. Inf. Syst. (2009)
25. Karvounarakis, G., et al.: Querying the semantic web with RQL. Computer Networks 42(5) (2003)
26. Langegger, A., Wöß, W., Blöchl, M.: A semantic web middleware for virtual data integration on the web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 493–507. Springer, Heidelberg (2008)
27. Lenzerini, M.: Data integration: A theoretical perspective. In: Symposium on Principles of Database Systems (2002)
28. Maedche, A., et al.: MAFRA - a MApping FRAmework for distributed ontologies. In: Knowledge Acquisition, Modeling and Management (2002)
29. Maedche, A., et al.: Managing multiple and distributed ontologies on the semantic web. VLDB J. 12(4) (2003)
30. Makris, K., Bikakis, N., Gioldasis, N., Tsinaraki, C., Christodoulakis, S.: Towards a mediator based on OWL and SPARQL. In: Lytras, M.D., Damiani, E., Carroll, J.M., Tennyson, R.D., Avison, D., Naeve, A., Dale, A., Lefrere, P., Tan, F., Sipior, J., Vossen, G. (eds.) WSKS 2009. LNCS, vol. 5736, pp. 326–335. Springer, Heidelberg (2009)
31. Miller, R.J., et al.: Schema mapping as query discovery. In: Very Large Data Bases (2000)
32. Motik, B., et al.: Bridging the gap between OWL and relational databases. J. Web Sem. 7(2) (2009)

33. Noy, N.F., Klein, M.C.A.: Ontology evolution: Not the same as schema evolution. Knowl. Inf. Syst. 6(4) (2004)
34. Noy, N.F., et al.: Making biomedical ontologies and ontology repositories work. IEEE Intelligent Systems 19(6) (2004)
35. Pan, A., et al.: The denodo data integration platform. In: Very Large Data Bases (2002)
36. Parreiras, F.S., et al.: Model driven specification of ontology translations. In: International Conference on Conceptual Modeling / the Entity Relationship Approach (2008)
37. Petropoulos, M., et al.: Exporting and interactively querying web service-accessed sources: The CLIDE system. ACM Trans. Database Syst. 32(4) (2007)
38. Polleres, A., Scharffe, F., Schindlauer, R.: SPARQL++ for mapping between RDF vocabularies. In: Chung, S. (ed.) OTM 2007, Part I. LNCS, vol. 4803, pp. 878–896. Springer, Heidelberg (2007)
39. Popa, L., et al.: Translating web data. In: Very Large Data Bases (2002)
40. Qin, H., Dou, D., LePendu, P.: Discovering executable semantic mappings between ontologies. In: Chung, S. (ed.) OTM 2007, Part I. LNCS, vol. 4803, pp. 832–849. Springer, Heidelberg (2007)
41. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
42. Raffio, A., et al.: Clip: a tool for mapping hierarchical schemas. In: SIGMOD Conference (2008)
43. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. 10(4) (2001)
44. Scharffe, F., et al.: Towards design patterns for ontology alignment. In: ACM Symposium on Applied Computing (2008)
45. Schmidt, M., et al.: Foundations of SPARQL query optimization. In: ICDT (2010)
46. Schmidt, M., et al.: SP^2Bench: A SPARQL performance benchmark. In: International Conference on Data Engineering (2009)
47. Serafini, L., Tamilin, A.: Instance migration in heterogeneous ontology environments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 452–465. Springer, Heidelberg (2007)
48. Shadbolt, N., et al.: The semantic web revisited. IEEE Intelligent Systems 21(3) (2006)
49. Sirin, E., et al.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2) (2007)
50. Thakkar, S., et al.: Composing, optimizing, and executing plans for bioinformatics web services. VLDB J. 14(3) (2005)
51. Uschold, M., Grüninger, M.: Ontologies and semantics for seamless connectivity. SIGMOD Record 33(4) (2004)
52. Yu, C., Popa, L.: Constraint-based XML query rewriting for data integration. In: SIGMOD Conference (2004)