On using database techniques for generating ontology mappings

Carlos R. Rivero
University of Sevilla, Spain
carlosrivero@us.es

Inma Hernández
University of Sevilla, Spain
inmahernandez@us.es

David Ruiz
University of Sevilla, Spain
druiz@us.es

Rafael Corchuelo
University of Sevilla, Spain
corchu@us.es

***Abstract*—In the Semantic Web, there are a variety of ontologies, which motivates the need for integrating them. Integration tasks rely on the use of relationships amongst the integrated ontologies, known as mappings. The literature reports on a number of techniques to automatically generate such mappings, unfortunately, the results are not suitable to perform integration tasks since they can produce incoherent data. The database community has devised techniques that automatically generate integration-enabling mappings, however, these techniques are not directly applicable to the semantic web context due to the inherent differences between ontologies and database models. In this paper, we present the differences between semantic web ontologies (RDF, RDFS and OWL) and nested relational schemata, which argue to develop new techniques to generate integration-enabling mappings. Furthermore, we analise the requirements to generate integration-enabling mappings in the context of the Semantic Web.**

***Index Terms*—Database technologies for semantic web; Ontology mediation.**

I. Introduction

Ontologies are suitable to solve semantic heterogeneity problems that can arise when two or more user communities share their knowledge [37]. However, in a distributed, evolving and open-world environment such as the Semantic Web, there are a variety of ontologies. This implies that using ontologies per se does not avoid semantic heterogeneity problems since they appear again when user communities share their different ontologies [12]. In this paper, we focus on semantic web ontologies, i.e., ontologies that are specified using RDF, RDFS or OWL [3].

As a result, the Semantic Web comprises heterogeneous and distributed ontologies, and there is a need to integrating them [4], [25]. Without an exception, integration tasks rely on the use of mappings, which are formulae represented in some logical formalism that relate the integrated ontologies [12]. Integration tasks include data translation and data integration. On the one hand, the data translation task is the process of moving the data that is stored at a number of source ontologies to a target ontology [9], [13], [14], [32]. On the other hand, the data integration task is the process of answering a query over a target ontology using only the data stored at the sources [17], [28], [40], [42]. In this paper, we focus on the data translation task.

Specifying hand-crafted mappings is tedious and error-prone since it leads the user to a frustrating trial-and-error loop: the user specifies the mappings and tests if their behaviour is correct; otherwise, the user has to rewrite the mappings and starts the loop again [30]. Note also that the costs of maintaining hand-crafted mappings are very high [43]. Therefore, automatic mapping generation relieves users from the burden of specifying hand-crafted mappings and reduces integration costs [23], [43]. There exists a large amount of literature that study how to generate these mappings automatically [8], [12], [20], [34]. Unfortunately, the mappings these techniques generate are not suitable to perform integration tasks since they can produce incoherent target data when considered in isolation or restrictions are not taken into account [23], [14], [32]. To solve these problems, the database community has devised techniques that automatically generate integration-enabling mappings based on correspondences, which are simple relationships amongst elements in the integrated database models [14], [32]. Note that our research focuses on nested relational schemata, which is a common abstraction for relational, XML and other hierarchical, set-oriented models [14]. Furthermore, nested relational techniques cannot be directly applied to the ontologies due to the inherent differences between ontologies and database models [16], [24], [26], [41].

In this paper, our goal is to support that techniques to generate integration-enabling mappings for nested relational schemata are not directly applicable to the semantic web case, due to inherent differences between database models and ontologies. As a result, we present a number of requirements for generating integration-enabling mappings in the semantic web context, and we analise them based on the inherent differences.

The paper is structured as follows: Section II describes the related work. Section III presents preliminaries regarding nested relational schemata and ontologies. In Section IV, we describe a number of inherent differences between nested relational schemata and semantic web ontologies. In Section V, we present a motivating example to discuss about why mapping generation techniques for nested relational are not directly applicable to semantic web ontologies, and the requirements for generating integration-enabling mappings in the context of the Sematic Web. Finally, Section VI summarises our main

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).

conclusions.

II. Related work

In the database context, correspondences are represented in multiple ways [32], [34]. Correspondences relate the most simple entities in the source and target models, e.g., a column in the relational model or an attribute in the nested relational model [34]. Furthermore, correspondences can relate one or more entities in the source with one or more entities in the target, so, they are of four types, namely: 1:1, n:1, 1:n, n:m.

Popa et al. [32] used one of the most simple form of correspondences: a logic equality between an attribute in the source and an attribute in the target (1:1 correspondences). The mapping system developed by Mecca et al. [21] handled a more general form of correspondences: they relate a number of source attributes with a target attribute via a transformation function (n:1 correspondences). Their mapping system also allows to restrict the way the correspondences must be combined. Raffio et al. [33] proposed correspondences not only between attributes but also between nodes.

Integration-enabling mapping generation techniques for nested relational schemata produce three different types of mappings, namely: basic, nested and laconic/core. The technique devised by Popa et al. [32] generates basic mappings, and it combines logically related correspondences in which referential integrity constraints and nesting are made explicit. Fuxman et al. [14] proposed to use nested mappings to improve the results of basic mappings, which have a number of problems such as the inefficiency in their execution or the redundancy in their specification. They generate basic mappings and compose nested mappings by rewriting them; this is more efficient since they factor out common expressions and reduce the number of basic mappings used to translate the data. Basic and nested mappings can produce redundant target instances, which motivated the research on the generation of laconic/core mappings that produce target instances with no redundancy [21], [38]. These techniques generate basic mappings and rewrite them to produce laconic/core mappings.

Basic, nested and laconic/core mappings are translated into a suitable query language, and the resulting mappings are known as query mappings. The data translation task consists of executing the query mappings over the source to produce instances of the target. The benefits of using query mappings are that the data translation process is simplified, making it more efficient and flexible: instead of relying on complex, ad-hoc programs that are difficult to create and maintain, thanks to query mappings, the database management system is used as the transformation engine [23]. Furthermore, database management systems incorporate a vast knowledge on query manipulation, from which it is derived that query mappings can be automatically optimised and paralelised so that data translation can perform as good as possible.

Finally, Alexe et al. [2] developed a benchmark for comparing and evaluating mapping generation techniques. The evaluation criteria include the scalability of the generated query mappings and the support of various typical scenarios.

Regarding ontologies, there are a number of proposals to represent the correspondences that are based on ontologies and rules. Ontology-based proposals describe them by means of a populated ontology [10], [19], e.g., the Semantic Bridge Ontology [19] allows to relate classes, properties, and individuals. Rule-based approaches describe the correspondences as bridge rules in a specific-purpose language [5], [27], e.g., C-OWL [5] allows bridge rules that relate classes in five ways: equivalent, into, onto, incompatible or compatible.

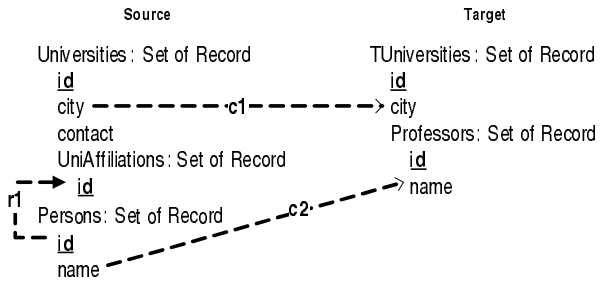
Maedche et al. [19] described a number of dimensions (aspects) for correspondences in the ontology context, namely: entity, cardinality, constraint, transformation and structural. The entity dimension establishes which entities are related, e.g., concepts, relations or attributes. The cardinality dimension covers how many entities are related ranging from 1:1 to n:m, however, the authors have found that 1:n and n:1 correspondences are sufficient in most general cases. The constraint dimension are conditions that must hold to perform an ontology data translation. The transformation dimension specifies how the values have to be translated during an ontology data translation. Finally, the structural dimension deals with how the correspondences are combined; these combinations include specialisation, abstraction, composition and alternative.

Regarding ontology mapping generation techniques, Choi et al. and Euzenat and Shvaiko [8], [12] surveyed a number of approaches to automatically generate ontology mappings. However, these ontology mappings are not suitable to perform integration tasks since they do not incorporate the model restrictions (e.g., domain or range or cardinality of properties) and do not combine logically related mappings. Early attempts that work with ontology mappings are Observer and Momis [7], [22], however, these systems focus on integration tasks and assume that mappings are available beforehand.

Regarding ontology data translation, current proposals rely on the use of ad-hoc techniques [19], reasoners [9], [36] or SPARQL query mappings [11], [29]. Regarding ad-hoc techniques, Maedche et al. [19], [18] built an execution engine in which the instances of the Semantic Bridge Ontology are evaluated to perform the data translation process. This engine is their first step towards developing a general translation technique for instances of the Semantic Bridge Ontology.

Regarding proposals based on reasoners, Dou et al. [9] perform data translation by means of a reasoner that needs a merged ontology that covers the source and target models. In this approach, the user specifies the mappings by using a first-order logic language and the reasoner is a first-order theorem prover that has been optimised for the data translation task. Serafini and Tamilin [36] work with correspondences between two classes, and their data translation process consists of reclassifying source instances into the target.

Regarding SPARQL queries, Euzenat et al. [11] presented preliminary ideas on the use of SPARQL queries to perform the data translation process. They focus on the lacks of SPARQL to work as a mapping query language [31]. Parreiras et al. [29] proposed a model-driven framework to



c1: Universities.city = TUniversities.city
c2: Persons.name = Professors.name
r1: Persons.id → UniAffiliations.id

m1: ∀ u · u ∈ Universities ⇒
∃ tu · tu ∈ TUniversities ∧ tu.city = u.city
m2: ∀ p, u, ua · p ∈ Persons ∧ u ∈ Universities ∧
ua ∈ u.UniAffiliations ∧ ua.id = p.id ⇒
∃ tu, prof · tu ∈ TUniversities ∧
prof ∈ tu.Professors ∧
tu.city = u.city ∧ prof.name = p.name

```
qm1:   <TUniversities>
       {
          for $u in $doc/Universities
          return
             <University>
                <id>generateId($u/city)</id>
                <city>$u/city</city>
             </University>
       }
       </TUniversities>
qm2: for $p in $doc/Persons,
...
```

Fig. 1. Examples of integration-enabling mappings in nested relational schemata

solve the automatic generation of query mappings: their pilot implementation translates hand-crafted mappings specified in OCL into SPARQL query mappings.

III. Preliminaries

In this section, we provide an example of nested relational schemata and semantic web ontologies. Furthermore, we provide examples of the concepts that we use throughout this paper.

In Figure 1, we present an example of two nested relational schemata, correspondences and mappings. The source schema comprises three nodes: $Universities$, $UniAffiliations$ and $Persons$. These nodes have a number of attributes, e.g., $city$ of $Universities$ or $name$ in $Persons$. Note that $UniAffiliations$ is nested into the $Universities$ node, furthermore, $r1$ is a referential integrity constraint that relates the id of $Persons$ and $UniAffiliations$. The target schema has $TUniversities$ and $Professors$ nodes, and $Professors$ is nested into $TUniversities$.

The relationships established between the source and the target schemata can be of three conceptual levels [15], namely:

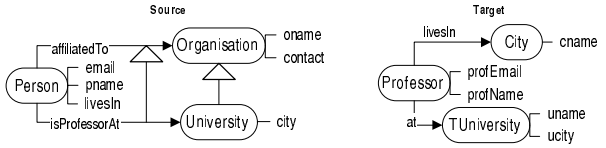


Fig. 2. Example of semantic web ontologies

- Correspondences are simple relationships amongst elements in the source and the target [32]. For example, in Figure 1, $c1$ and $c2$ are two correspondences relating $city$ and $name$ attributes in the source and target schemata. Correspondences can be given by means of an automatic matching tool [12], [34], by the user with the help of a graphical tool [1], [33], or using patterns [35].
- Integration-enabling mappings are formulae represented in some logical formalism, which combine logically related correspondences in which the inherent restrictions entailed by the modelling language are made explicit [15]. For example, in Figure 1, $r1$ is a referential integrity restriction of the model and $m1$ and $m2$ are two integration-enabling mappings, which are created based on correspondences $c1$ and $c2$, and restriction $r1$.
- Query mappings are the translation of integration-enabling mappings into a suitable query language such as SQL, XQuery or SPARQL [11], [14], [43]. Examples of query mappings include $qm1$ and $qm2$ in Figure 1.

In Figure 2, we present two semantic web ontologies that are intended to represent the same concepts as the nested relational schemata shown in Figure 1. Languages to describe semantic web ontologies allow to specify classes and properties that relate them. In our example, the classes are represented as oval shapes, e.g., in Figure 2, $Person$ or $Organisation$ are classes of the source ontology. The classes are similar in spirit to nodes in nested relational schemata.

Regarding properties, they have a domain and a range: the domain consists of a set of classes (possibly one), and the range can be either a set of classes (possibly one) or a basic data type. OWL distinguishes two types of properties: object and data properties. Object properties relate two classes, e.g., in the source ontology in Figure 2, $affiliatedTo$ means that a $Person$ (domain) is affiliated to one or more $Organisations$ (range). Data properties relate a class with a constant of a simple type, e.g., in the source ontology in Figure 2, $pname$ is the name of the $Person$ (domain), which is a String (range) but we omit it for the sake of simplicity.

Note that, when a semantic web ontology is populated, the data does not reflect all the implicit implications entailed by the modelling language. For instance, in the source ontology in Figure 2, if an instance of $University$ exists, this instance has implicitly the $Organisation$ type. Reasoners are used to make this knowledge explicit for a populated ontology [39]. This task is mandatory for some applications, e.g, when SPARQL is used to query an ontology, it is mandatory to make the knowledge explicit (in the query or in the model)

because SPARQL only deals with RDF and does not implement RDFS/OWL semantics. However, there are other query languages that implement RDFS/OWL semantics, e.g., SeRQL [6].

IV. Nested relational schemata versus semantic web ontologies

In this section, we present the inherent differences we have found between the nested relational schemata and semantic web ontologies, namely:

D1 Structure and data: data, in nested relational schemata, cannot exist before a model is devised for this data, e.g., in the source schema in Figure 1, it is not possible to have an instance of $Universities$ without the schema. However, in the Semantic Web, data exists before it is modeled since data in this context amounts to pre-existing web resources, e.g., it is possible to have an URI that represents a person, but this data has no type until it is explicitly modeled to be of type $Person$ in the source ontology in Figure 2. Furthermore, several (even conflicting) models can exist for the same data.

D2 Weak existential relations: in nested relational schemata, the semantics of an attribute is that if a node instance exists, then there must exist an attribute instance for each attribute in the schema; if the value is not known, then it is null. However, in semantic web ontologies, data properties are similar to attributes but, due to the weak existential relations, if the value of a data property instance is unknown, then the data property instance does not exist. For instance, attribute $city$ of node $Universities$ in the source schema in Figure 1 has to exist and, if its value is unknown, then it is null. However, in the source ontology in Figure 2, an instance of data property $oname$ that is related to an instance of $Organisation$ only exists if the data property instance has a known value.

Furthermore, in nested relational schemata, referential integrity constraints and nesting are the mechanisms to relate nodes, which implies related nodes must exist. For example, in the source schema in Figure 1, the nesting of $UniAffiliations$ in $Universities$ implies that an instance of $UniAffiliations$ exists if and only if and instance of $Universities$ exists (but not conversely). A similar behaviour occurs with the referential integrity constraint $r1$, which implies that, if an instance of $Persons$ exists, then an instance of $UniAffiliations$ must exist (but not conversely).

In the semantic web context, object properties can be modelled as referential integrity constraints or nesting but, due to the weak existential relations, instances of object properties do not enforces the mandatory existence of instances of their domains or ranges. For example, we cannot assume that an instance of $Person$ and an instance of $University$ in the source ontology in Figure 2 are related by property $isProfessorAt$, since isolated instances of $Person$ and $University$ are allowed without no object properties instances involved.

D3 Subclasses and subproperties: in semantic web ontologies, when a class is specialised into a subclass, an instance of the subclass has also the type of the superclass, i.e., the final instance has both the type of the class and the subclass. In the source ontology in Figure 2, the specialisation of classes is represented as a white triangle with a line connecting the subclass, e.g., $University$ is subclass of $Organisation$, which means that any instance of $University$ is also an instance of $Organisation$.

When a property is specialised into a subproperty, the domain and range classes of the property are implicitly added to the domain and the range of the subproperty. In the instance level, two instances related by the subproperty are automatically related by the superproperty. In the source ontology in Figure 2, the specialisation of properties is represented with a white triangle that relates two properties, e.g., $isProfessorAt$ is a subproperty of $affiliatedTo$. If an instance of $Person$ and an instance of $University$ are related by $isProfessorAt$, they are also related by the $affiliatedTo$ property. Furthermore, $affiliatedTo$ has $Person$ as its domain, and its range is $Organisation$ and $University$.

Note that, in nested relational schemata, there is no equivalent to subclasses and subproperties. There are a number of extensions to database models that take classes and subclasses into account. However, none of them deals with subproperties or multi-type instances (see below).

D4 Instances of multiple types: in semantic web ontologies, one instance can be of multiple types without a relationship amongst them, e.g., the source ontology in Figure 2 can be populated with an instance that is both a $Person$ and $University$. Note that this is not a mistake since a type is a classifier for an existing web resource. For instance, http://en.wikipedia.org/wiki/Edsger_W._Dijkstra can be classified as both a $Person$ resource and $University$ resource.

In nested relational schemata, an instance is a set of attribute values that specify a concrete node in the schema, e.g., in the source schema in Figure 1, $Universities$ is specified by the values of id, $city$ and $contact$ attributes. Therefore, in nested relational schemata, it is impossible to have multiple types since the concept of multiple types is not supported.

D5 Properties are globally defined: in semantic web ontologies, properties are not local to a certain class but they are global to the whole ontology, e.g., assume that in the source ontology in Figure 2, we change the $pname$ and $oname$ properties by a new one called

$name$, this entails that data property instances of $name$ can have instances of $Person$, $Organisation$ or both as domain.
In nested relational schemata, attributes are local to each node, e.g., in the source schema in Figure 1, each instance of $Persons$ has an attribute id, and each instance of $Universities$ has another attribute id but they are totally different, without any relation between them.

D6 URIs: in semantic web ontologies, every class, property or instance is identified by an URI, i.e., a Uniform Resource Identifier, which can be an URL (web address) or some other type of unique identifier. In nested relational schemata, unique identifiers have to be made explicit by means of primary keys. For example, in the source schema in Figure 1, the id attributes in $Universities$, $UniAffiliations$ and $Persons$ are different.

V. Discussion

In this section, we argue that existing techniques to generate integration-enabling mappings for nested relational schemata are not applicable to the semantic web context. Furthermore, we present the requirements that such a technique should fulfill.

Assume that we use the ontologies in Figure 2 as a data translation scenario. Regarding weak existential relations, assume that we establish two correspondences that relate the source and the target of the type data property to data property (data to data), which are similar to attribute correspondences in database models:

dd1 (data to data): pname = profName
dd2 (data to data): city = ucity

We argue that data to data correspondences are not sufficient to generate ontology mappings. Recall that integration-enabling mapping generation techniques for nested relational models are based on correspondences between attributes and the inherent restrictions of these models, i.e., referential integrity constraints and nesting (cf. Figure 1). However, ontology languages do not impose any inherent restrictions since ontology properties are weak existential relations. Therefore, if a mapping generation technique for nested relational schemata is applied to our example in Figure 2 with $dd1$ and $dd2$ correspondences, then, mappings that include object properties $isProfessorAt$ and at will not be generated since the existence of instances of these object properties is not mandatory.

To solve this problem, Maedche et al. [19] introduced two new types of correspondences: object property to object property and class to class. What follows is an example of these correspondences:

oo1 (object to object): isProfessorAt = at
cc1 (class to class): Person = Professor

Another solution is to restrict the cardinality of the properties, which can be seen as enforcing the existence of an instance property. For example, if we establish that the cardinality of property $isProfessorAt$ in our source ontology is exactly one (by means of the "cardinality" construct in the OWL language), this implies that every instance of $Person$ is associated with one instance of $University$ by $isProfessorAt$.

Note that cardinality restrictions are only allowed by the OWL language but not by the RDFS language. This motivates the need to explicitly indicate that a property in an RDFS ontology is mandatory, i.e., the existence of an instance of the property is mandatory only when an instance of its domain or range exists. In our example, we make property $isProfessorAt$ mandatory by enforcing the existence of an instance of $University$, i.e., we only consider instances of $Person$ that are related to one or more instances of $University$ by $isProfessorAt$.

Regarding subclasses and subproperties and the instances of multiple types, nested relational models do not support neither of them, therefore, it is not possible to apply mapping generation techniques over ontologies with subclasses, subproperties or instances of multiple types.

After this discussion, we identify a number of requirements for the generation of integration-enabling mappings, namely:

R1 New types of correspondences: data property to data property correspondences are not sufficient to generate integration-enabling mappings because properties are weak existential relations (cf. difference D2). It is necessary at least three types of correspondences: class to class, data property to data property, and object property to object property.

R2 Explicit mandatory properties: in semantic web ontologies, properties are optional if cardinality is not restricted (cf. difference D2). However, in a concrete scenario, we may wish to have into account a property as mandatory.

R3 Subclasses and subproperties: every class and property can be specialised into a subclass or a subproperty (cf. difference D3). An integration-enabling mapping generation technique must deal with these specialisations.

R4 Multiple types, domains and ranges: instances can be of multiple types (cf. difference D4). It is also possible that the properties are multi-domain and/or -range, which is related to the global definition of the properties (cf. differences D5).

VI. Conclusions

In the bibliography, there are several techniques to automatically generate ontology mappings, which relieves users from the burden of specifying hand-crafted mappings and helps reduce integration costs. However, these resulting mappings are not suitable to perform data translation since they can lead to incoherent data. Mappings that are suitable to perform integration tasks are known as integration-enabling mappings.

The database community has devised several techniques to automatically generate integration-enabling mappings, which, as far as we know, have not been studied yet in the semantic

web context. Furthermore, database techniques to generate integration-enabling mappings are not directly applicable to the semantic web context due to the inherent differences between database models and ontologies.

In this paper, we argue that more research on the generation of integration-enabling mappings is required in the semantic web context. We analise a number of differences between nested relational schemata and semantic web ontologies. Finally, we present a number of requirements to generate integration-enabling mappings.

REFERENCES

[1] B. Alexe, L. Chiticariu, R. J. Miller, D. Pepper, and W. C. Tan. Muse: a system for understanding and designing mappings. In *SIGMOD Conference*, pages 1281–1284, 2008.

[2] B. Alexe, W. C. Tan, and Y. Velegrakis. STBenchmark: towards a benchmark for mapping systems. *PVLDB*, 1(1):230–244, 2008.

[3] G. Antoniou and F. van Harmelen. *A Semantic Web Primer, 2nd Edition*. The MIT Press, 2008.

[4] P. A. Bernstein and L. M. Haas. Information integration in the enterprise. *Commun. ACM*, 51(9):72–79, 2008.

[5] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing ontologies. *J. Web Sem.*, 1(4):325–343, 2004.

[6] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In *International Semantic Web Conference*, pages 54–68, 2002.

[7] G. Cabri, F. Guerra, M. Vincini, S. Bergamaschi, L. Leonardi, and F. Zambonelli. Momis: Exploiting agents to support information integration. *Int. J. Cooperative Inf. Syst.*, 11(3):293–314, 2002.

[8] N. Choi, I.-Y. Song, and H. Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34–41, 2006.

[9] D. Dou, D. V. McDermott, and P. Qi. Ontology translation on the Semantic Web. *J. Data Semantics*, 2:35–57, 2005.

[10] J. Euzenat. An API for ontology alignment. In *International Semantic Web Conference*, pages 698–712, 2004.

[11] J. Euzenat, A. Polleres, and F. Scharffe. Processing ontology alignments with SPARQL. In *CISIS*, pages 913–917, 2008.

[12] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[13] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[14] A. Fuxman, M. A. Hernández, C. T. H. Ho, R. J. Miller, L. Popa, and P. Papotti. Nested mappings: Schema mapping reloaded. In *VLDB*, pages 67–78, 2006.

[15] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *SIGMOD Conference*, pages 805–810, 2005.

[16] G. Karvounarakis, A. Magkanaraki, M. Scholl, D. Plexousakis, S. Alexaki, V. Christophides, and K. Tolle. Querying the Semantic Web with RQL. *Computer Networks*, 42(5):617–640, 2003.

[17] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[18] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the Semantic Web. *VLDB J.*, 12(4):286–302, 2003.

[19] A. Maedche, B. Motik, R. Volz, and N. Silva. Mafra: A mapping framework for distributed ontologies. In *EKAW*, pages 235–250, 2002.

[20] M. Mao, Y. Peng, and M. Spring. An adaptive ontology mapping approach with neural network based constraint satisfaction. *J. Web Sem.*, 2010.

[21] G. Mecca, P. Papotti, and S. Raunich. Core schema mappings. In *SIGMOD Conference*, pages 655–668, 2009.

[22] E. Mena, A. Illarramendi, V. Kashyap, and A. P. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

[23] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In *VLDB*, pages 77–88, 2000.

[24] B. Motik, I. Horrocks, and U. Sattler. Bridging the gap between OWL and relational databases. *J. Web Sem.*, 7(2):74–89, 2009.

[25] N. F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[26] N. F. Noy and M. C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440, 2004.

[27] M. J. O'Connor, H. Knublauch, B. N. Grosof, S. W. Tu, W. E. Grosso, M. Dean, and M. A. Musen. Supporting rule system interoperability on the Semantic Web with SWRL. In *International Semantic Web Conference*, pages 974–986, 2005.

[28] C. R. Osuna, D. Ruiz, R. Corchuelo, and J. L. Arjona. SPARQL Query Splitter: query translation between different contexts. In *JISBD*, pages 320–323, 2009.

[29] F. S. Parreiras, S. Staab, A. Winter, and S. Schenk. Model driven specification of ontology translations. In *ER*, pages 484–497, 2008.

[30] M. Petropoulos, A. Deutsch, Y. Papakonstantinou, and Y. Katsis. Exporting and interactively querying web service-accessed sources: The CLIDE system. *ACM Trans. Database Syst.*, 32(4), 2007.

[31] A. Polleres, F. Scharffe, and R. Schindlauer. SPARQL++ for mapping between RDF vocabularies. In *OTM Conferences (1)*, pages 878–896, 2007.

[32] L. Popa, Y. Velegrakis, R. J. Miller, R. Fagin, and M. A. Hernández. Translating web data. In *VLDB*, pages 598–609, 2002.

[33] A. Raffio, D. Braga, S. Ceri, P. Papotti, and M. A. Hernández. Clip: a tool for mapping hierarchical schemas. In *SIGMOD Conference*, pages 1271–1274, 2008.

[34] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[35] F. Scharffe, J. Euzenat, and D. Fensel. Towards design patterns for ontology alignment. In *SAC*, pages 2321–2325, 2008.

[36] L. Serafini and A. Tamilin. Instance migration in heterogeneous ontology environments. In *ISWC/ASWC*, pages 452–465, 2007.

[37] N. Shadbolt, T. Berners-Lee, and W. Hall. The Semantic Web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[38] B. ten Cate, L. Chiticariu, P. G. Kolaitis, and W. C. Tan. Laconic schema mappings: Computing the core with sql queries. *PVLDB*, 2(1):1006–1017, 2009.

[39] H. J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.*, 3(2-3):79–115, 2005.

[40] S. Thakkar, J. L. Ambite, and C. A. Knoblock. Composing, optimizing, and executing plans for bioinformatics web services. *VLDB J.*, 14(3):330–353, 2005.

[41] M. Uschold and M. Grüninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[42] C. Yu and L. Popa. Constraint-based XML query rewriting for data integration. In *SIGMOD Conference*, pages 371–382, 2004.

[43] C. Yu and L. Popa. Semantic adaptation of schema mappings when schemas evolve. In *VLDB*, pages 1006–1017, 2005.