Generic Schema Matching, Ten Years Later

Philip A. Bernstein
Microsoft Corporation
philbe@microsoft.com

Jayant Madhavan
Google Inc.
jayant@google.com

Erhard Rahm
University of Leipzig
rahm@informatik.uni-leipzig.de

ABSTRACT

In a paper published in the 2001 VLDB Conference, we proposed treating generic schema matching as an independent problem. We developed a taxonomy of existing techniques, a new schema matching algorithm, and an approach to comparative evaluation. Since then, the field has grown into a major research topic. We briefly summarize the new techniques that have been developed and applications of the techniques in the commercial world. We conclude by discussing future trends and recommendations for further work.

1. INTRODUCTION

Schema matching is the problem of generating correspondences between elements of two schemas. A *schema* is a formal structure that represents an engineered artifact, such as a SQL schema, XML schema, entity-relationship diagram, ontology description, interface definition, or form definition. A *correspondence* is a relationship between one or more elements of one schema and one or more elements of the other. For example, the correspondences in Figure 1 identify columns that represent the same concepts in the two relational schemas. Often, the relationship is one-to-one, but sometimes it is not, such as Author corresponding to LastName and FirstName in Figure 1. We say that a correspondence has *semantics* if it constrains the instances of the related schema elements. The common default semantics for one-to-one correspondences is that the instances of two related elements are equal.

There are many applications that require schema matching. In the database field, it is usually the first step in generating a program or view definition that maps instances of one schema into instances of another. For example, it arises in object-to-relational mappings, data warehouse loading, data exchange, and mediated schemas for data integration. In knowledge-based applications, such as life sciences applications and the semantic web, it arises in the alignment of ontologies. For example, it may be used to align gene ontologies or anatomical structures. In health care, it may arise in the alignment of patient records and other medical reports. In web applications, it may be used to align product catalogs. In e-commerce, it may be used to align message formats representing business documents, such as orders and invoices.

This paper recaps the contributions of our VLDB 2001 paper about schema matching [45], summarizes developments since then, and suggests problems that would benefit from further work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 37th International Conference on Very Large Data Bases, August 29th - September 3rd 2011, Seattle, Washington.

Proceedings of the VLDB Endowment, Vol. 4, No. 11

Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

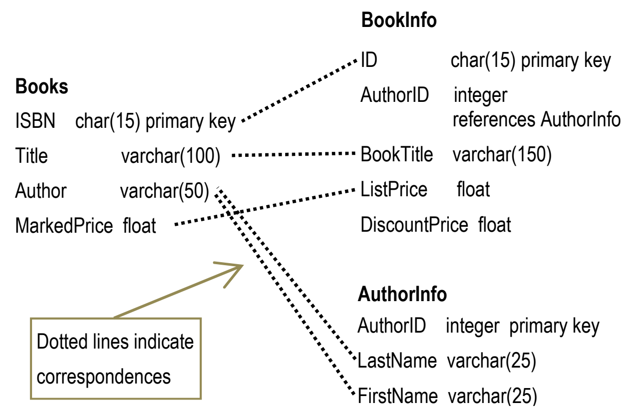

Figure 1: Schema matching is the problem of generating correspondences that identify related elements in two schemas.

2. CONTRIBUTIONS IN VLDB 2001 [45]

Twelve years ago, when we embarked on work in this area, we noticed that schema matching techniques were developed as part of a variety of applications. The techniques were often similar, even when the applications were not. We concluded that the field might move faster and the results might be more reusable if schema matching were studied as a separate topic, independently of the applications that use it. This recommendation was the first contribution of [45].

We then surveyed the literature to identify these common techniques. This resulted in a taxonomy of schema matching techniques, which was the second contribution of [45]. We extended this taxonomy into a survey paper, published later that year in [63]. The taxonomy has often been used as a standard for categorizing subsequent schema matching techniques.

Our third contribution was a new schema matching algorithm, called Cupid, which combined a number of techniques: linguistic matching, structure-based matching, constraint-based matching, and context-based matching. Most of the later approaches to schema matching have used this hybrid matcher approach, which leverages different criteria to arrive at suggested correspondences.

We concluded with an experimental comparison of Cupid with two other algorithms that were reported in the literature, namely MOMIS [6] and DIKE [58]. This was the first such comparison we know of. Such experimental comparisons have become a feature of most of the later work on schema matching.

In summary, our 2001 paper posed schema matching as a problem that could be studied in isolation. It gave a baseline of known techniques. And given the inherently heuristic nature of solutions, it suggested an approach to evaluate those solutions based on experiments. As the references in [45] attest, we were by no means the first to work on schema matching. However, we defined a framework for research on this topic that enabled many others to follow.

3. SCHEMA MATCHING TECHNIQUES

There are now two books on schema matching [5][26] and two surveys [63][68], so there is little point in our repeating such a survey. However, to give the reader a feel for the scope of the schema matching field, we list many of the known techniques here. We start with techniques that were known in 2001 and that we discussed in [45]:

- Linguistic matching – based on an element's name or description, using stemming, tokenization, string and substrings matching, and information retrieval techniques.
- Using auxiliary information – based on thesauri, acronyms, dictionaries, and mismatch lists.
- Instance-based matching – schema elements are regarded as similar if their instances are similar, based on statistics, metadata, or trained classifiers.
- Structure-based matching – schema elements are similar if they appear in similarly-structured groups, have similar relationships, or have (paths of) relationships to similar elements.
- Constraint-based matching – based on data types, value ranges, uniqueness, nullability, and foreign keys.
- Rule-based matching – based on matching rules that are expressed in first-order logic.
- Hybrid-matching – as explained in the previous section.

Since 2001, many other techniques have been developed. These include algorithms that use new types of information. For example:

- Graph matching – based on comparing the relationships between elements in the schema graphs by, for example, either fixed-point computations on a similarity propagation graph [53], or probabilistic constraint satisfaction algorithms [22].
- Usage-based matching – based on analyzing database query logs for hints about how users relate schemas, e.g., by equating elements in join clauses [25]. Taxonomy paths can be matched by finding web pages that represent the paths and then analyzing keyword-query logs to determine if the pages are accessed via similar query distributions [55].
- Document content similarity – where instances of a schema element are grouped into a document that is then matched with other such documents based on the information retrieval measure TF-IDF (term frequency times inverse document frequency) [44][49].
- Document link similarity – where concepts in two ontologies are regarded as similar if the entities referring to those concepts are similar [42].

Strategies have been proposed to flexibly combine multiple matching algorithms and to scale gracefully to compare large schemas. For example:

- Workflow-like strategies to independently or sequentially execute matchers and to combine their results [12][19][67].
- Self-tuning match workflows – where for a given match task or domain of match tasks, a tuner selects the match components to be combined and/or assigns values to the various parameters that affect how component match results are combined [24][43][44].
- Early search space pruning – where a fast matcher is used to eliminate unlikely matches from consideration so that a manageably-small number of elements can be matched using more expensive and accurate techniques [23][57].
- Partition-based matching – where to reduce the space of possible matches, the input schemas are partitioned followed by partition-wise matching [20][39][73].
- Parallel matching – where different steps of the matching algorithm are run in parallel or different partitions of the schemas are matched in parallel [34].
- Optimizations for large schemas such as using string matching optimizations [40], pre-collecting predecessors and children of each element to avoid repeated traversal [2], and using space-efficient similarity matrices [12].

Approaches have been proposed where multiple schemas in a domain are collectively matched. For example:

- Reuse-based matching – where matches between schema fragments are harvested from validated mappings and used as auxiliary information to help future match tasks in the same domain [20][46][65].
- Holistic matching – where a single mediated schema is constructed for a domain by aligning elements of a large corpus of schemas, such as web forms covering a particular domain. Similar element names appearing in the same schema are regarded as mismatches [37][38][66][69].

Strategies have been proposed to incorporate user interaction and feedback in the matching process. For example:

- GUI support to interactively inspect and correct computed correspondences [3][11][16][31].
- Incremental matching – where given a user-selected element of one schema, the matcher calculates the best match or matches (top-k) in the other schema [11].
- Top-k matching – where instead of computing a complete mapping between two schemas, the matcher computes the top-k matches of each element of one schema to elements of the other schema [11][32].
- Collaborative, wiki-like user involvement to provide, improve, and reuse mappings [50][72].

Finally, algorithms have been proposed that extend the semantics of matches beyond that of simple correspondences. For example:

- Semantic tagging – where correspondences are tagged with semantic relationships, such as equality, containment, disjointness, and unknown. [33][35] [48].
- Conditional tagging – where correspondences are refined to be valid only for certain values of another element. For example, if productType = "book" then Invoice.Code = ISBN [14][33].

4. SCHEMA MATCHING TOOLS

Most of the listed techniques have been implemented in a large number of tools for schema and ontology matching [26][62]. Figure 2 shows a comparative overview of selected tools: Cupid [45], COMA++ [3][19][20], ASMOV [40], Falcon-AO [39], RiMON [44], AgreementMaker [16], OII Harmony [67]. Most

		Cupid	COMA++	Falcon-AO	RiMON	ASMOV	Agreement-Maker	OII Harmony
Year of introduction		2001	2002/2005	2006	2006	2007	2007	2008
Input	*relational*	√	√	-	-	-	-	√
schemas	*XML*	√	√	-	-	-	√	√
	ontologies	-	√	√	√	√	√	√
OAEI participation		-	√	√	√	√	√	-
Comprehensive GUI		-	√	(√)	?	?	√	√
Matchers	*linguistic*	√	√	√	√	√	√	√
	structure	√	√	√	√	√	√	√
	instance	-	√	-	√	√	√	-
Use of external dictionaries		√	√	?	√	√	√	√
Schema partitioning		-	√	√	-	-	-	-
Parallel matching		-	-	-	-	-	-	-
Dynamic matcher selection		-	-	-	√	-	-	-
Mapping reuse		-	√	-	-	-	-	-

Figure 2 Comparison of selected match tools (based on [62]).

recent prototypes support match workflows and the combined use of different linguistic, structural and instance-based matchers. External dictionaries such as synonym lists or thesauri are commonly used to improve linguistic matching. GUI support is often provided, albeit still with limitations [31]. A few systems are able to match both schemas and ontologies [3][16][67]. As indicated in Figure 2, advanced techniques such as schema partitioning, parallel matching, mapping reuse and self-tuning capabilities (e.g., a dynamic selection of matchers for a given match task) are still only supported to a limited extent in current match prototypes.

Match tools have been intensively evaluated but typically under different conditions and for smaller match problems [4][18]. For ontology matching, the Ontology Alignment Evaluation Initiative (OAEI) organizes yearly contests that include some larger problems, e.g., to match web directories or medical ontologies (http://oaei.ontologymatching.org). The systems participating in the OAEI contest have significantly improved over the years but still struggle with larger problems [27]. For schema matching and mapping, a comparable benchmark effort is still missing.

Semi-automatic schema matching is also increasingly supported in commercial middleware tools, in particular for XML schemas or relational database schemas. Systems such as Altova MapForce, IBM Infosphere, Microsoft BizTalk Server and SAP Netweaver provide a GUI-based editor for manual mapping specification with some support for automatic determination of match candidates, e.g., based on approximate name matching. However, most of the more recently proposed match techniques have not yet been incorporated in commercial mapping solutions.

5. USING MATCH RESULTS AS-IS

Even the best schema matching algorithms make many mistakes, especially fully-automatic algorithms where there is no human designer in the loop. Despite these errors, some applications can use schema matching results as-is. This is especially the case when a best-effort matching is satisfactory or when the matches contribute only implicitly to the results of some end-user task. For example, consider the following two scenarios for automatically filling out HTML forms.

First, most of today's browsers offer automatic form-filling, e.g., personal data such as name and address prior to a purchase. This can be modeled as a task where the schema of the underlying web-form is being matched to a model of user data that is stored in the browser. The user expects the browser to make a best-effort attempt at filling in personal details, which the user confirms before submitting the form for processing.

Figure 3 Mappings between domain models and form inputs can be used to automatically fill out HTML forms.

Second, schema mappings have been proposed as a means of accessing the content that lies behind HTML forms [47][61]. A deep-web crawler can work as follows: When the crawler encounters an HTML form, it can identify the domain that the form belongs to, and then match the inputs of the form to elements in the previously-computed mediated schema for that domain (see Figure 3). It can then generate form submissions by constructing URLs using sample values for the inputs (based on known values for the elements in the mediated schema). The resulting pages are added to the index of the search engine. The matching results in this case are intermediate results of a multi-step process. End-users are unlikely to know or care about the quality of the match result, except insofar as it affects how the crawler exploits the underlying website.

6. APPLYING MATCH TO MODEL MANAGEMENT

For most of the applications summarized in Section 1, schema matching is just one step in a multi-step process. That multi-step process involves other operators that manipulate schemas and mappings, such as schema merging and mapping composition. This recognition was actually the starting point for our research into schema matching. In [8] and [9], we proposed a set of such operators under the name "model management". We then embarked on a systematic study of these operators. Since nothing much can be done until the first mapping is created, it was logical that we started our investigation of operators with schema matching. In fact, our first algorithmic result about one of the operators was our paper "Generic Schema Matching" [45]. Since then, there has been a lot of progress on other operators in addition to match, which is summarized in [10].

Most data integration and data transformation applications, such as those in Section 1, need to construct executable mappings—ones that represent transformations of instances. Since match algorithms produce correspondences, not semantic relationships, the natural next step is to enrich those correspondences with semantics [54]. Often this is a two-step process (Section 3.1 of [10]). The first step is to generate semantics in the form of constraints that relate parts of the instances of one schema to parts of the instances of another schema. Such constraints may not be functions, in which case they are not executable. In this case, a second step is needed to translate the semantic relationships into functions [51] via the operator TransGen.

Depending on the application, the resulting mapping may need to undergo further manipulation. Suppose we match schemas S and T and then generate a semantic mapping between them. We might want to *merge* S and T into a single schema that covers both of them, for example, to represent a mediated schema. This can be done by the merge operator, which takes as input two schemas and a mapping between them and returns a merged schema with mappings between the merged schema and the two input schemas [15][59][60][64].

Suppose we are using the mapping between S and T as a data transformation that translates data from S's format into T's format. If one of the schemas T in a mapping is modified, generating T′, then we need to update the mapping between S and T to one between S and T′. We can do this by *composing* the mappings S-T and T- T′ [30][36], yielding a mapping T-T′ between S and T′ [7][28][71].

Other model management operators are Diff (which finds the difference between mappings) and Extract (the complement of Diff) [52], and Invert, which reverses the direction of a unidirectional mapping [28][29].

For most practical applications, all of the model management operators manipulate mappings that have semantics—except for the match operator which has a special role. First the match operator computes correspondences and then, building on these correspondences, the other operators develop and manipulate mappings that have semantics.

7. FUTURE TRENDS

Since 2001, there has been a growing realization that matching is not a one-of task. For example, in data integration, as new data sources become available, they are mapped to a single mediated schema. In e-commerce, message formats of new business partners have to be mapped to message formats that interface to existing business processes. It is natural to expect that with each subsequent task to match within a given domain or to a given schema, the effort required to construct the mapping should decrease, while the quality of the mapping should increase.

For a given vertical domain, such as product catalogs or patient records, there are many possible schemas. These schemas exhibit common patterns, which can be used to improve the results of a schema matching algorithm. Most of the early approaches to schema matching encoded this domain knowledge as constraints or heuristics that were baked into the algorithm. The encoded constraints were developed by a designer with intimate knowledge of the domain.

A more flexible approach was introduced in [21]. It showed that new mappings to a mediated schema can be learned from known mappings to that schema. Machine-learning algorithms were used to train models for elements in the mediated schema using known mappings. The models were then applied to the elements in new schemas to map them to the same mediated schema. The approach was extended in [17] to learn complex expressions in addition to just correspondences. It was further extended in [46] to show that models can be trained from known mappings in a domain and applied to match two completely new schemas in the same domain.

Much of the value of mappings is in the semantic expressions that are developed from the initial correspondences. It is therefore important to reuse those expressions, not simply generate correspondences based on learned models. An early approach in [19] proposed reusing a validated mapping fragment F by matching the source and target of the schemas to be matched with the source and target of F. This introduces several related problems. First, there is the question of how to partition a schema into fragments, whose validated mappings can be reused. Second, a repository is needed to store and provide access to validated mappings [1]. Third, there is the combinatorial problem of finding possible matches of each mapping in the library to the many positions where it might fit in the source and target of the schemas to be matched. One attempt is discussed in [20]. More work along these lines is needed.

Despite this progress in mapping reuse, little of the technology has made it into commercial offerings.

The availability of large numbers of schemas on the web makes the holistic matching approach quite appealing. Collective schema matching was proposed in [37] and applied in [38] to match the inputs in HTML forms. Many schemas (i.e., forms) that are known to be in a given domain are collectively analyzed to infer a single mediated schema for that domain. Then a generative model is learned for the domain based on the assumption that each distinct schema is simply a different representation of a subset of a single underlying domain schema. Subsequent work has extended this clustering approach to accommodate more complex mappings between HTML forms [70]. These approaches have thus far been restricted to form matching where the schemas are small, with just a few, well-understood underlying concepts in the domain.

In most schema matching scenarios, there is a human in the loop. Therefore, it is important to have excellent graphical support for viewing mappings [31]. For example, since large schemas cannot be viewed on a single screen, it is beneficial to partition them into fragments that can be matched independently, to the extent possible. Matching tools also need to offer better support for the mapping process. For example, users need help in remembering which schema elements they have examined during the match process and what was learned by that examination, such as promising and specious candidates.

We see an increasing convergence of schema matching and entity resolution approaches, i.e., matching at the metadata level and matching at the instance level. Most recent schema and ontology matching prototypes include instance-based matchers [61] that derive the similarity of schema elements from the similarity or overlap of element instances. Entity resolution, i.e., the identification of semantically corresponding entities or instances, can benefit from the semantic categorization of entities within ontologies and the provision of ontology mappings. For example, the organization of products or product offers within product catalogs helps to restrict product matching between different sources to corresponding or closely related product categories, based on a pre-determined ontology mapping between the product catalogs. Link discovery to interconnect sources in the so-called web of linked data [13][56] is an area where such semantic entity resolution approaches are needed and applicable due to the broad availability of ontologies.

8. CONCLUSION

In this paper, we briefly summarized generic schema matching developments since we published our 2001 paper that introduced the subject [45]. We listed published techniques, how published techniques are used, and future trends.

There seem always to be new sources of information available to new schema matching techniques and clever ways of combining existing techniques. In this sense, the problem of schema matching is inherently open-ended. Thus, the schema matching field is still a vibrant one, with many opportunities for researchers and tool developers to move it forward.

9. ACKNOWLEDGMENTS

We thank the many researchers who have collaborated with us over the years, helping us learn many of the lessons summarized in this paper. They include Eddie Churchill, Hong-Hai Do, AnHai Doan, Alon Halevy, Sabine Massmann, Sergey Melnik, Michalis Petropoulos, and Christoph Quix.

10. REFERENCES

[1] Alexe, B. M. Gubanov, M. A. Hernandez, H. Ho, J.-W. Huang, Y. Katsis, L. Popa, B. Saha, and I. Stanoi. Simplifying Information Integration: Object-Based Flow-of-Mappings Framework for Integration. Proc. BIRTE, 108–121. Springer, 2009.

[2] Algergawy, A., E. Schallehn, and G. Saake: Improving XML schema matching performance using Prüfer sequences. Data Knowl. Eng. 68(8), 728-747, 2009.

[3] Aumueller, D., H.H. Do, S. Massmann, and E. Rahm: Schema and ontology matching with COMA++. Proc. SIGMOD, demo paper, 906-908, 2005.

[4] Bellahsene, Z., A. Bonifati, F. Duchateau, and Y. Velegrakis: On Evaluating Schema Matching and Mapping. In: Z. Bellahsene, A. Bonifati, E.Rahm (eds), Schema Matching and Mapping, Springer, 2011.

[5] Bellahsene, Z., A. Bonifati, and E. Rahm (editors), *Schema Matching and Mapping*, Springer, 2011.

[6] Bergamaschi, S., S. Castano, and M. Vincini: Semantic Integration of Semistructured and Structured Data Sources. SIGMOD Record 28(1), 54-59, 1999.

[7] Bernstein, P.A., T. J. Green, S. Melnik, and A. Nash: Implementing mapping composition. VLDB J. 17(2), 333-353, 2008.

[8] Bernstein, P.A., L.M. Haas, M. Jarke, E. Rahm, and G. Wiederhold: Panel: Is Generic Metadata Management Feasible? Proc. VLDB, 660-662, 2000.

[9] Bernstein, P.A., A.Y. Halevy, and R. Pottinger: A Vision of Management of Complex Models. SIGMOD Record 29(4), 55-63, 2000.

[10] Bernstein, P.A. and S. Melnik: Model Management 2.0: Manipulating Richer Mappings. Proc. SIGMOD, 1-12 2007.

[11] Bernstein, P.A., S. Melnik, and J.E. Churchill: Incremental schema matching. Proc. VLDB, demo paper, 1167-1170, 2006.

[12] Bernstein, P.A., S. Melnik, M. Petropoulos, and C. Quix: Industrial-Strength Schema Matching. ACM SIGMOD Record 33(4), 38-43, 2004.

[13] Bizer, C., T. Heath, and T. Berners-Lee: Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst. 5(3), 1-22, 2009.

[14] Bohannon, P. E. Elnahrawy, W. Fan, and M. Flaster: Putting context into schema matching. Proc. VLDB, 307-318, 2006.

[15] Chiticariu, L., P. G. Kolaitis, and L. Popa: Interactive generation of integrated schemas. Proc. SIGMOD, 833-846, 2008.

[16] Cruz, I.F., F.P. Antonelli, and C. Stroe: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB 2(2), demo paper, 1586-1589, 2009.

[17] Dhamankar, R., Y. Lee, A-H. Doan, A.Y. Halevy, and P. Domingos: iMAP: Discovering Complex Mappings between Database Schemas. Proc. SIGMOD, 383-394, 2004.

[18] Do, H.H., S. Melnik, and E. Rahm: Comparison of Schema Matching Evaluations. In: Web, Web-Services, and Database Systems, Springer LNCS 2593, 221-237, 2003.

[19] Do, H.H. and E. Rahm: COMA – A System for Flexible Combination of Schema Matching Approaches. Proc. VLDB, 610-621, 2002.

[20] Do, H.H. and E. Rahm: Matching large schemas: Approaches and evaluation. Inf. Syst. 32(6), 857-885, 2007.

[21] Doan, A-H., P. Domingos, and A.Y. Halevy: Reconciling the Schemas of Disparate Data Sources: A Machine-Learning Approach. Proc. SIGMOD, 509-520, 2001.

[22] Doan, A-H., J. Madhavan, P. Domingos, and A. Y. Halevy: Learning to Map between Ontologies on the Semantic Web. Proc. WWW, 662-673, 2002.

[23] Ehrig M., and S. Staab: Quick ontology matching. Proc. Int. Conf. Semantic Web (ICSW), Springer LNCS 3298, 683-697, 2004.

[24] Ehrig M., S. Staab, and Y. Sure: Bootstrapping Ontology Alignment Methods with APFEL. Proc. Int. Conf. Semantic Web (ICSW), Springer LNCS 3729, 186-200, 2005.

[25] Elmeleegy, H., M. Ouzzani, and A.K. Elmagarmid: Usage-Based Schema Matching. Proc. ICDE, 20-29, 2008

[26] Euzenat, J. and P. Shvaiko, *Ontology Matching*, Springer, 2007.

[27] Euzenat, J., A. Ferrara, C. Meilicke, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Šváb-Zamazal, V. Svátek and C. Trojahn dos Santos: Final Results of the Ontology Alignment Evaluation Initiative 2010. Proc. 5th ISWC workshop on Ontology Matching, 2010.

[28] Fagin, R.: Inverting schema mappings. ACM Trans. Database Syst. 32(4), Article 25, 2007.

[29] Fagin, R., P.G. Kolaitis, L. Popa, and W.-C. Tan: Quasi-inverses of schema mappings. ACM Trans. Database Syst. 33(2), Article 11, 2008.

[30] Fagin, R., P.G. Kolaitis, L. Popa, and W.-C. Tan: Schema Mapping Evolution Through Composition and Inversion. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping, Springer, 191-222, 2011.

[31] Falconer, S.M. and N.F. Noy: Interactive Techniques to Support Ontology Matching. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping, Springer, 29-52, 2011.

[32] Gal, A.: Managing uncertainty in schema matching with top-k schema mappings. J. Data Semantics 6, 90-114, 2006.

[33] Gal, A.: Enhancing the capabilities of attribute correspondences. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping, Springer, 53-74, 2011.

[34] Gross, A., M. Hartung, T. Kirsten, and E. Rahm: On Matching Large Life Science Ontologies in Parallel. Proc. 7th Int. Conf. Data Integration in the Life Sciences (DILS), Springer LNCS 6254, 35-49, 2010.

[35] Giunchiglia F., P. Shvaiko, and M. Yatskevich: Semantic schema matching. Proc. OTM Conf. (CoopIS), 347-365, 2005.

[36] Hartung, M., J. Terwilliger, and E. Rahm: Recent advances in schema and ontology evolution. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping, Springer, 149-190, 2011.

[37] He, B. and K. C.-C. Chang: Statistical Schema Matching across Web Query Interfaces. Proc. SIGMOD, 217-228, 2003.

[38] He, H., W. Meng, C.T. Yu, and Z. Wu: Automatic integration of Web search interfaces with WISE-Integrator. VLDB J. 13(3), 256-273, 2004.

[39] Hu, W., Y. Qu, and G. Cheng: Matching large ontologies: A divide-and-conquer-approach. Data Knowl. Eng. 67(1), 140-160, 2008.

[40] Jean-Mary, Y.R., E.P. Shironoshita, and M.R. Kabuka: Ontology matching with semantic verification, J. Web Semantics 7(3):235–251, 2009.

[41] Koudas, N., A. Marathe, and D. Srivastava: Flexible String Matching Against Large Databases in Practice. Proc. VLDB, 1078-1086, 2004.

[42] Lambrix, P., H. Tan, and W. Xu: Literature-based Alignment of Ontologies. Proc. ICSW Ontology Matching (OM) workshop, 2008.

[43] Lee, Y., M. Sayyadian, A. Doan, and A. Rosenthal: eTuner: tuning schema matching software using synthetic scenarios. VLDB Journal 16(1), 97-122, 2007.

[44] Li, J., J.Tang, Y. Li, and Q. Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Trans. Knowl. Data Eng. 21(8), 1218-1232, 2009.

[45] Madhavan, J., P. A. Bernstein, and E. Rahm: Generic Schema Matching with Cupid. Proc. VLDB, 49-58, 2001.

[46] Madhavan, J., P.A. Bernstein, A. Doan, and A.Y. Halevy: Corpus-based Schema Matching. Proc. ICDE, 57-68, 2005.

[47] Madhavan, J., D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Y. Halevy: Google's Deep Web crawl. PVLDB 1(2), 1241-1252, 2008.

[48] Magnani, M., N. Rizopoulos, P. McBrien, and D. Montesi: Schema integration based on uncertain semantic mappings. Proc. ER, Springer LNCS 3716, 31-45, 2005.

[49] Massmann, S. and E. Rahm: Evaluating Instance-based Matching of Web Directories. Proc. WebDB, 2008.

[50] McCann, R., W. Shen, and A. Doan: Matching schemas in online communities: A web 2.0 approach. Proc. ICDE, 110-119, 2008.

[51] Melnik, S., A. Adya, and P.A. Bernstein: Compiling Mappings to Bridge Applications and Databases. ACM Trans. Database Syst. 33(4), Article 22, 2008.

[52] Melnik, S., P.A. Bernstein, A.Y. Halevy, and E. Rahm: Supporting Executable Mappings in Model Management. Proc. SIGMOD, 167-178, 2005.

[53] Melnik, S., H. Garcia-Molina, and E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. Proc. ICDE, 117-128, 2002.

[54] Miller, R.J., L.M. Haas, and M.A. Hernández: Schema Mapping as Query Discovery. Proc. VLDB, 77-88, 2000.

[55] Nandi, A. and P.A. Bernstein: HAMSTER: Using Search Clicklogs for Schema and Taxonomy Matching. PVLDB 2(1), 181-192, 2009.

[56] Parundekar, R., C.A. Knoblock, and J.L. Ambite: Linking and Building Ontologies of Linked Data. Proc. Int. Semantic Web Conference, Springer LNCS 6496, 598-614, 2010.

[57] Peukert, E., H. Berthold, and E. Rahm: Rewrite Techniques for Performance Optimization of Schema Matching Processes. Proc. EDBT, 433-464, 2010.

[58] Palopoli, L., G. Terracina, and D. Ursino: The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses. Proc. ADBIS-DASFAA, 108-117, 2000.

[59] Pottinger, R.: Mapping-Based Merging of Schemas. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping, Springer, 223-252, 2011.

[60] Pottinger, R. and P.A. Bernstein: Merging Models Based on Given Correspondences. Proc. VLDB, 826-837, 2003.

[61] Raghavan, S. and H. Garcia-Molina: Crawling the Hidden Web. Proc. VLDB, 129-138, 2001.

[62] Rahm, E.: Towards Large-Scale Schema and Ontology Matching. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping, 3-28, Springer, 2011.

[63] Rahm, E. and P.A. Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB J. 10(4), 334-350, 2001.

[64] Raunich, S. and E. Rahm: ATOM: Automatic Target-driven Ontology Merging. Proc. ICDE, demo paper, 276-279, 2011,

[65] Saha, B., I. Stanoi, and K.L. Clarkson. Schema covering: a step towards enabling reuse in information integration. Proc. ICDE, 285-296, 2010.

[66] Saleem, K., Z. Bellahsene, and E. Hunt: PORSCHE: Performance Oriented SCHEma mediation. Inf. Syst. 33(7-8),637-657, 2008.

[67] Seligman, L., P. Mork, A.Y. Halevy, K. Smith, M,J. Carey, K. Chen, C. Wolf, J. Madhavan, A. Kannan, and D. Burdick: OpenII: An Open Source Information Integration Toolkit, Proc. SIGMOD, 1057-1060, 2010.

[68] Shvaiko, P. and J. Euzenat: A Survey of Schema-based Matching Approaches. Journal on data Semantics, IV, 146-171, 2005.

[69] Su, W., J. Wang, and F.H. Lochovsky: Holistic Schema Matching for Web Query Interfaces. Proc. EDBT, 77-94, 2006.

[70] Wu, W., C.T. Yu, A.-H. Doan, and W. Meng: An Interactive Clustering-based Approach to Integrating Source Query interfaces on the Deep Web. Proc. SIGMOD, 95-106, 2004.

[71] Yu, C. and L. Popa: Semantic Adaptation of Schema Mappings when Schemas Evolve. Proc. VLDB, 1006-1017, 2005.

[72] Zhdanova, V. and P. Shvaiko: Community-Driven Ontology Matching. Proc. ESWC, 34-49, 2006.

[73] Zhong, Q., H. Li, J. Li, G. Xie, J. Tang, L. Zhou, and Y. Pan: A Gauss Function Based Approach for Unbalanced Ontology Matching. Proc. SIGMOD, 669-680, 2009.