On using high-level structured queries for integrating deep-web information sources

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

Rafael Z. Frantz
Uniju, Brasil
rzfrantz@unijui.edu.br

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

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

***Abstract*—The actual value of the Deep Web comes from integrating the data its applications provide. Such applications offer human-oriented search forms as their entry points, and there exists a number of tools that are used to fill them in and retrieve the resulting pages programmatically. Solution that rely on these tools are usually costly, which motivated a number of researchers to work on virtual integration, also known as metasearch. Virtual integration abstracts away from actual search forms by providing a unified search form, i.e., a programmer fills it in and the virtual integration system translates it into the application search forms. We argue that virtual integration costs might be reduced further if another abstraction level is provided by issuing structured queries in high-level languages such as SQL, XQuery or SPARQL; this helps abstract away from search forms. As far as we know, there is not a proposal in the literature that addresses this problem. In this paper, we propose a reference framework called IntegraWeb to solve the problems of using high-level structured queries to perform deep-web data integration. Furthermore, we provide a comprehensive report on existing proposals from the database integration and the Deep Web research fields, which can be used in combination to address our problem within the previous reference framework.**

***Index Terms*—Internet and emerging technologies; Semantic Web.**

I. Introduction

The Deep Web is composed of millions of applications that provide valuable data, which is usually served by querying search forms coded in HTML [4], [15], [52]. There are a number of studies in the bibliography about the Deep Web, which state a growth in the number of deep-web applications. Bergman's report [4] estimated 200 000 applications in 2001, Chang et al. [15] estimated 307 000 applications in 2004 and, finally, Madhavan et al. [52] estimated 25 million of deep-web applications in 2007.

Our research focus on the usage of high-level structured queries to integrate the deep-web data, which may help reduce the cost of a deep-web data integration solution.

To integrate the deep-web data is crucial but challenging since its data is behind search forms [50], which are designed by and for users and they do not have formally-defined semantics. Virtual integration (also known as metasearch [16], [37]) is a technique to perform deep-web data integration [52]. Virtual integration approaches provide a unified search form for a specific domain, e.g., travels, hotels or flights. This search form is built by merging search forms of the same or related domains [35], [37]; the user fills it in, and the system translates the unified search form to fill the application search forms in.

Virtual integration approaches issue queries through search forms by means of the field values of the unified search form, which has proved to reduce integration costs [16], [37]. Increasing the abstraction level using high-level structured query languages may indeed help reduce integration costs. In the virtual integration approaches, the unified search form abstracts away from the actual applications, dealing with the query capabilities of the application search forms. High-level structured queries abstract away even from the unified search form, which has also its own query capabilities and do not have formally-defined semantics: the developer of a integration solution is only concerned with developing appropriate queries, which are posed over the solution. Note that the unified search form allows more specific queries than keyword-based query interfaces, and high-level structured languages allow even more specific and complex queries.

In this paper, we propose to use high-level structured queries to perform deep-web data integration, and report on a reference framework called IntegraWeb that combines results from the database integration and the Deep Web research fields. Specifically, database integration techniques provide the architecture of the solution. Deep-web virtual integration approaches are used to deal with the search forms of the deep-web applications. Finally, both deep-web surfacing and virtual integration approaches retrieve data pages and, in combination with information extraction and ontologising techniques, extract structured data from these pages.

This paper is organised as follows: Section II presents the research efforts made on the database integration and the Deep Web research fields. In Section III, we present IntegraWeb and we survey the state of the art in deep-web data integration. We conclude with Section IV, which presents our conclusions.

II. Research on data integration

In the last 10-15 years, data integration has been a very active research field [33]. The first approaches to data integration were related to database and, in next years, these approaches have been converging to the Web. Note that high-level structured queries can be specified in a number of

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).

languages, e.g., SQL or XQuery [7] are used in the context of database, and SPARQL [61] in the Semantic Web arena.

Database integration techniques focus on providing unified access to a number of applications. Each application has one or more models of its data, which is commonly called schemes. Schemes are comprised of schema attributes, which are the properties of the schema, e.g., the departure date of a flight. Amongst the schemes there are a number of semantic mappings ("semantic glue"), which are formulae that provide the semantic relationships between the schema attributes [51], [54].

The existing approaches in the database integration research field are Mediators [26], [33], Peer Data Management Systems (PDMS) [20] and PayGo systems [52], [68].

Mediators offer a mediated schema to the schemes of the integrated applications (known as application schemes), and they need wrappers to provide access to them. Mediators deal with the translation of the user query by means of the semantic mappings. The user query is posed over the mediated schema and it is translated into some queries over the suitable application schemes. Finally, the data of the different applications is combined to give an answer to the user. TSIMMIS [26] is an example of a Mediator.

Peer Data Management Systems (or PDMS) are the next step in database integration systems after Mediators. The mediated schema in Mediators is actually a bottleneck in the process [31], [32]. Instead of having a unique mediated schema, the applications in PDMS have their own schema and there are mappings between these schemes. An example of a PDMS is Piazza [30].

PayGo systems are the next step after PDMS and they have a gradually increasing presence in the database integration community [22], [50], [52], [68]. PayGo systems are inspired in the concept of Dataspaces [22], [29] that do not require full semantic integration of the applications to provide integration services. These systems offer one schema for each application (as in PDMS) or a mediated schema (as in Mediators) but there is a key difference: the uncertainty. Instead of full semantic mappings, PayGo systems use probabilistic mappings and schemes. An example of PayGo systems is UDI [68].

The existing techniques that are used to have access to the Deep Web are surfacing [2], [43], [53], [65], [71] and virtual integration [16], [37].

Deep-web surfacing approaches (also known as crawling [2], [43], [53], [65], [71]) provide automatic access to the Deep Web. These approaches expose deep-web pages behind search forms and index them in search engines. These techniques precompute the most relevant form submissions for the selected search forms. These systems do not have to cope with the problem of building schemes, instead of this, the challenge is to automatically generate relevant form submissions that retrieve significant data pages. Google's Deep Web Crawl is an example of a deep-web surfacing system [53].

Virtual integration approaches work similarly to Mediators. They offer a unified search form that is created by using the search forms of the integrated applications, which are usually of the same or related domains. The user fills the unified search form in and it is used to fill the search forms of the different deep-web applications in. After submitting a filled form, a result page is obtained, and it is typically a list of links to data pages, e.g., a list of books in Amazon. Finally, data pages have to be retrieved. An example of a deep-web virtual integration system is MetaQuerier [16].

III. IntegraWeb

IntegraWeb is our contribution to combine the database integration and the Deep Web techniques. IntegraWeb is a reference framework that allows to integrate deep-web data using a high-level structured query language, such as SQL, XQuery or SPARQL.

In Figure 1, we present an example of deep-web data integration that consists of a unique entry point, which models information about travels (virtual schema), and it integrates two application schemes that models information about flights and hotels (application schemes). The Query Processing task takes the user query over the virtual schema as input and translates it into a number of queries to the application schemes, in the example, the user wishes to start the travel on October, 5 and the price must not exceed 3500€. This query is translated into a number of queries for flights and hotels.

The Search Form Processing task takes one application query as input and it calculates a number of search form submissions that has to performed to answer the application query. In our example, for flights, an origin and a destination airport need to be filled in; note that if the application query specifies the name of the city, it has to be transformed into the airport(s) of this city. The Deep Web Accessing task deals with the extraction of deep-web application instances by filling the search forms in, navigating through the result pages and extracting and giving semantics to the data pages. Finally, the instances have to be filtered and compiled in both Search Form Processing and Query Processing tasks.

In the next sections, we describe the processes performed by each task in the IntregraWeb reference framework, and we survey the most related literature.

A. Query processing

The Query Planner (cf. Figure 2) takes a high-level structured user query as input, this query has to be expressed in terms of a virtual schema, and it generates an execution plan by using the mappings involving other schemes. The execution plan consists of a number of queries, each of which is related to one application schema only. The execution of the queries can be ordered (i.e., the results of a query are used by another query) or parallel (i.e., independent queries). The Plan Executor takes an execution plan as input and iterates until it is finished. In each iteration, a query over an application schema is processed. Finally, the Query Compiler combines the intermediate instances retrieved from the applications using the execution plan and returns the results to the user. These intermediate instances may need filtering.

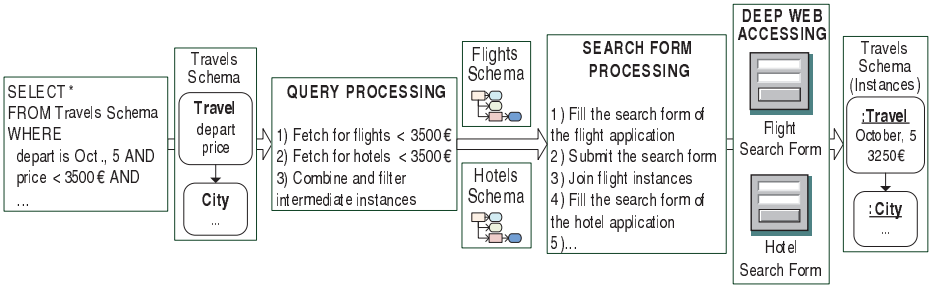

Fig. 1. An example of deep-web data integration

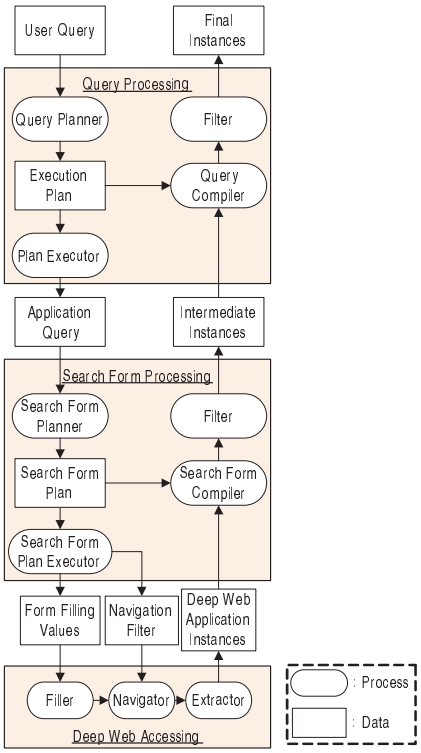

Fig. 2. The IntegraWeb reference framework

In the Query Processing task of our example (cf. Figure 1), the execution plan could be parallel, if the applications do not have any dependency between them, or could be ordered and it retrieves firstly flight instances and then hotel instances, e.g., the departure flight departs at October, 5 and arrives at the destination city at October, 6, the idea is to book the hotel the same day we arrive at the destination city. Therefore, in this example, the arrival date at the hotel needs to be October, 6 but note that we start the journey on October, 5. There exists a dependency between the flight application and the hotel application. To the best of our knowledge, there is not a proposal that takes this problem into account, i.e., the dependency between two (or more) application schemes.

In this module, any database integration technique is suitable. When using Mediators, the system is configured having a mediated schema and a number of application schemes. In Mediators, mappings are defined mainly using two techniques: Global-as-View (GaV) and Local-as-View (LaV) [33]. In GaV, the mediated schema is defined in terms of views over application schemes and, in LaV, application schemes are defined in terms of views over the mediated schema [28], [45]. Another technique is GLaV [24] that combines the advantages of LaV and GaV. A query posed over a GaV system is answered by query unfolding [9] and, in LaV systems, by query answering [28]. Another technique to answer a query over a mediated schema is to approximate it to the application schemes [14].

In other systems, such as PDMS or PayGo, the semantic mappings are specified by using GaV, LaV, GLaV. Generating those mappings is a labour-intensive and error prone task, so it is needed tools that help user to build and maintain them or tools that generate them automatically. The Clio system is an example of a tool that helps users to specify their mappings [27].

The Clio system requires a user to devise the mappings. The process of inferring such mappings automatically is commonly referred to as schema matching. Rahm and Bernstein [66] surveyed the schema matching techniques in 2001, a more recent survey is presented in [21]. Schema matching is a very active research field, but it is still an open problem because of an unavoidable consequence of ambiguity in the meaning of the data to be integrated [5]. Therefore, the intervention of a human is needed.

According to Bernstein and Melnik [5], the solution is to raise the level of abstraction in which mappings are specified, and they propose the model management technique that supports working with mappings between schemes in a high abstraction level. Another challenge in the semantic mappings field is to work with uncertainty, an approach is probabilistic semantic mappings and PayGo systems support them [68].

Finally, the Semantic Web research field is facing up the problem of querying distributed applications, and there are some approaches that translate a user query into a number of queries that are issued to multiple applications. Quilitz and Leser [63] or Langegger et al. [44] divide a SPARQL user query into a number of queries that are issued to multiple SPARQL endpoints. User queries are issued to a global schema that is comprised of multiple endpoints, which are defined as views over the global schema. Note that in the Semantic Web context, the recommendation of the W3C for the high-level

structured query language is SPARQL [61].

B. Search Form processing

The Search Form Planner (cf. Figure 2) takes an application query as input, which is expressed in terms of one application schema, and generates a search form plan that consists of a number of search form submissions, i.e, a number of form filling values which are taken from the query that is being analysed, and a filter for each submission, this filter is applied during navigation.

The Search Form Plan Executor takes a search form plan and feeds the Filler with form filling values and the Navigator with the appropriate filter. Note that query values could need some transformations to perform form filling. There can exist schema attributes that do not correspond to any fields in the search form. The constraints in a query that refer to those attributes must be grouped and applied as a post-filter to the results the search form returns. This process is performed by the Search Form Compiler, which is also responsible for combining the result instances retrieved by the Extractor, using the search form plan.

Existing approaches model a search form as a parameterised view over the application schema [59]. In this context, to have an answer for an application query, we can use the techniques for answering queries using views [28] (cf. Section III-A). A key concept in this module is query feasibility: it analyses whether a query can be issued over a schema without executing it. This analysis avoids a trial and error process in which the system poses a query and executes it until reach a suitable query. Petropoulos et al. [60] presented CLIDE, a user interface for building SQL queries over a set of parameterised views. CLIDE, besides query building, warns users when a query over the selected views is not feasible.

Pan et al. report on [59] a generic framework for representing query capabilities. This framework analyses the feasibility of SQL queries over applications that include deep-web applications. An implementation of this framework and a recap on its main drawbacks is presented in [67].

In deep-web virtual integration, some approaches deal with the possibility of filling more than one search form in, having only one query. For example, if we have a search form of flights in which we have to select a price range between 0-1500€, 1500-5000€ and >5000€, and a user query is of the form "price $<$ 3500€", we need to fill two search forms in, one with 0-1500€ and another with 1500-5000€. In this last case, a filter of flights whose price is less than 3500€ is needed. Zhang et al. [76] developed a translation technique, from the unified search form to application search forms, that allows multiple form submissions.

C. Deep Web accessing

The Filler (cf. Figure 2) takes a number of form filling values as input and fills the application search form in. After filling, this search form is submitted and the resulting page is sent to the Navigator. The Navigator takes the result page and classifies it into three categories: error, data or list. An error page finishes the process, a data page is returned immediately and, if the result page is a list page, the Navigator navigates to the data pages by clicking on suitable links. Some optimisations can be done in this process, for example, if we are looking for flights whose cost is less than 3500€, we can avoid clicking on flights whose price is higher. To perform this optimisation, the Navigator uses the filters given as input by the Search Form Plan Executor.

Finally, the resulting data pages are returned to the Extractor. The Extractor takes these data pages as input and produces intermediate instances. This task is performed by extracting information of the web page and giving semantics to this information.

Regarding form filling, it is needed a search form model to give semantics to search forms, which are designed by and for users. Deep-web approaches use different types of search form models [2], [36], [43], [55], [65], [75]. The first step to generate a search form model is to identify labels, i.e., text strings that give users an intuition about the semantics of a form field [2], [36], [39], [43], [55], [58], [65], [75].

There are three different approaches to identify form field labels automatically, and they rely on the idea that the label positions in a search form have significant semantic information. In textual identification [36], [39], [43], the HTML code of a search form is used to extract field labels. These techniques rely on the idea that analysing HTML code approximately captures the visual layout. In layout position techniques [2], [55], [65], [75], besides the HTML code, physical layout is used to extract field labels. In machine learning approaches, a variety of algorithms are combined to identify field labels [58].

The next step is to identify hidden database attributes. Search forms issue queries to deep-web databases whose structure is hidden partially, since a quick glance at the results of submitting a search form can reveal some of their attributes. These techniques extract hidden database attributes using only a search form. Some approaches, after label extraction, identify fields as attributes [2], [43], [65]. Other approaches allow attributes comprised of form fields with their associated labels [36], [55], [75], [76]. Kushmerick uses a machine learning algorithm to perform attribute extraction [41]. Some form models offer more information than labels and attributes, e.g., field order in form filling [25], mandatory fields [69], attribute logic relationships or attribute units [36], and search form query capabilities [69], [75].

Search form query capabilities are the different modes of querying a search form, e.g., a search form of books accepts queries by title, which is mandatory, author, publication year or any combination of them. The proposals that rely on an advanced search form model extract the search form query capabilities [69], [75], [76]. Shu et al. [69] extracts them by issuing predefined queries through search forms that help detect mandatory fields. Zhang et al. [75], [76] extract hidden database attributes, operators that are applied to these attributes, and their ranges. Attributes are combined by conjunctive queries because this is enough to capture the

query capabilities of most deep-web applications.

Regarding navigation, an important task is to classify the page that results from a search form submission. This result page can be an error page, a data page or a list page; in the last case, it is necessary to navigate through the page links to access to the data pages, which have to be also classified. Classifiers of web pages use a number of features that can be from the textual content, structural content or the visual layout [62], [71]. Note that the classifiers usually work with the significant portion of a web page that is the piece that results from removing miscellaneous headers and footers.

Textual content techniques study features such as predefined text patterns [47], [64], [69], e.g., "No matches" for error pages or "Showing 1 - 20 of 50 000" for list pages. Structural content approaches use the HTML tag tree to extract the features [6], [10], [71], e.g., the position of a label in the tag tree have significant meaning, if the label is present in the significant portion of the web page it has to be considered. Visual layout techniques use the features from the visual representation rendered by a web browser [2], [39], [64], [75], e.g., a web page with a number of images and some text at the right side of each image can be a list page that shows title pages of books and their corresponding information.

Caverlee and Liu [10] combines textual and structural techniques to perform web page classification. This approach fetches for areas in web pages that can be used to answer a user query. At the beginning, they cluster web pages according to their structural layout and, after this process, they filter each cluster using textual features such as the size of the web page.

A web page can contain some features that are missing, misleading, or unrecognisable for various reasons, e.g., the web page contains a number of large images or Flash objects but too little textual content. In these cases, the neighbours of the web page are used to supply supplementary information for the classification process [62]. A neighbour of a web page A is another web page B that has a link of the form $A \rightarrow B$ or $A \leftarrow B$ whose distance is one. Distances greater than one can be used to perform the neighbour analysis [62]. Besides the links between web pages, there are a number of approaches that use artificial links such as textual similarities between the pages or the pages that co-occur in top query results [62].

Navigation patterns are used to navigate through links [43], [56], [57], they define common navigation paths that are repeated in several deep-web applications. Another technique that can also be applied to navigate through links is focused crawling [11]. Focused crawling techniques or those proposals that use navigation patterns to optimise navigation processes rely on a blind search, which results in unnecessary clicks that lead to uninteresting pages. This argues for a more intelligent method to navigate through deep-web applications.

One challenge is to avoid these excessive clicks by identifying summary information of interest in list pages, and extracting this information so that uninteresting links are not clicked. Summary information has to be detected using record extraction techniques [1], [40], [46], [74], [77], and data extraction can be performed by information extractors. Montoto et al. [56], [57] presented a workflow language that allows to define a task to not perform a blind search.

Regarding extracting, there are four types of information extraction: hand-crafted, supervised, little supervised and unsupervised. Hand-crafted extractors rely on the user to provide the extraction rules [17], [34], [48]. Supervised extractors use a set of labeled documents to learn extraction rules using induction procedures [8], [23], [42]. Little supervised extractors work the same as supervised ones but with just one labeled document [13], [38]. Unsupervised extractors learn the extraction rules without requiring a set of labeled documents [18], [49], [72], [74]. Turmo et al. [70] survey information extractors that work on natural language data pages and Chia-Hui Chang et al. [12] survey information extractors that work on structured and semi-structured data pages. Information extraction techniques extract pieces of text from data pages; later they must be endowed with semantics by means of ontologisers [3], [19], [73].

IV. Conclusions

Deep-web virtual integration approaches reduce the integration costs by providing a unified search form, which abstracts away from the actual search forms. We argue that increasing the abstraction level may help reduce the cost of a deep-web data integration solution even further. This new abstraction consists of working with high-level structured queries that are posed over the integration solution: high-level structured queries allow to abstract away from the deep-web application search forms, and even from the unified search forms. Also, high-level structured languages allows more specific and complex queries than the unified search forms or keyword-based query interfaces.

We believe that this new abstraction level avoids the developer to be concerned with the details of the integration process, such as the execution plan that has to be used to retrieve and integrate the data, or the search form submissions that have to be performed to answer a query. The key problem when dealing with (application or unified) search forms is that they have some query capabilities, which have to be taken into account when posing queries over it. Furthermore, the search forms are human-oriented and they have not formally-defined semantics.

In this paper, we propose IntegraWeb as a reference framework to perform deep-web data integration by means of high-level structured queries. IntegraWeb emerges as the synergy between the research efforts made on database integration and the Deep Web, and it uses the advances of both research fields in combination. To the best of our knowledge, there is not a proposal that uses high-level structured query languages to integrate deep-web data. Furthermore, we survey the state of the art in both research fields and how the existing techniques have to be used in our reference framework.

Database integration research field has focused on the integration of data stored in different applications. The existing techniques go from a unique entry point by which the user query the system (Mediators), to a totally distributed system

in which each application has its data model and the user query any of them (PDMS and PayGo systems). Also, the mappings between the different data models can be totally exact at the beginning of the integration process (Mediators and PDMS), or they can be basic and evolve during this process (PayGo systems).

The research on the Deep Web has focused on accessing and integrating web data. The main challenge on developing these techniques is to have into account the search forms, i.e., deep-web applications deliver their data by means of a search form that have to be filled in by the user. In this research field, there are two main approaches: retrieving and indexing data pages to allow search engines to have access to them (surfacing), or retrieving data pages, extracting data from these pages and aggregating the results, to deliver the data to the user as if only one deep-web application had been queried (virtual integration).

REFERENCES

[1] M. Álvarez, A. Pan, J. Raposo, F. Bellas, and F. Cacheda. Extracting lists of data records from semi-structured web pages. *Data Knowl. Eng.*, 64(2):491–509, 2008.

[2] M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro. Crawling the Content Hidden Behind Web Forms. In *ICCSA*, pages 322–333, 2007.

[3] J. L. Arjona, R. Corchuelo, D. Ruiz, and M. Toro. From Wrapping to Knowledge. *IEEE Trans. Knowl. Data Eng.*, 19(2):310–323, 2007.

[4] M. K. Bergman. The Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, 7(1), 2001.

[5] P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *SIGMOD Conference*, pages 1–12, 2007.

[6] L. Blanco, V. Crescenzi, and P. Merialdo. Structure and Semantics of Data-IntensiveWeb Pages: An Experimental Study on their Relationships. *J. UCS*, 14(11):1877–1892, 2008.

[7] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. Technical report, W3C, 2007.

[8] M. E. Califf and R. J. Mooney. Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.

[9] D. Calvanese, D. Lembo, and M. Lenzerini. Survey on methods for query rewriting and query answering using views. Technical Report D1.R5, Universitá di Roma, 2001.

[10] J. Caverlee and L. Liu. QA-Pagelet: Data Preparation Techniques for Large-Scale Data Analysis of the Deep Web. *IEEE Trans. Knowl. Data Eng.*, 17(9):1247–1262, 2005.

[11] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

[12] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A Survey of Web Information Extraction Systems. *IEEE Trans. Knowl. Data Eng.*, 18(10):1411–1428, 2006.

[13] C.-H. Chang and S.-C. Kuo. OLERA: Semisupervised Web-Data Extraction with Visual Support. *IEEE Intelligent Systems*, 19(6):56–64, 2004.

[14] K. C.-C. Chang and H. Garcia-Molina. Approximate query mapping: Accounting for translation closeness. *VLDB J.*, 10(2-3):155–181, 2001.

[15] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured Databases on the Web: Observations and Implications. *SIGMOD Record*, 33(3):61–70, 2004.

[16] K. C.-C. Chang, B. He, and Z. Zhang. Toward Large Scale Integration: Building a MetaQuerier over Databases on the Web. In *CIDR*, pages 44–55, 2005.

[17] V. Crescenzi and G. Mecca. Grammars Have Exceptions. *Inf. Syst.*, 23(8):539–565, 1998.

[18] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: automatic data extraction from data-intensive web sites. In *SIGMOD Conference*, page 624, 2002.

[19] H. Davulcu, S. Vadrevu, and S. Nagarajan. OntoMiner: automated metadata and instance mining from news websites. *IJWGS*, 1(2):196–221, 2005.

[20] A. Doan and A. Y. Halevy. Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine*, 26(1):83–94, 2005.

[21] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer, 2007.

[22] M. J. Franklin, A. Y. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Record*, 34(4):27–33, 2005.

[23] D. Freitag. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, 39(2/3):169–202, 2000.

[24] M. Friedman, A. Y. Levy, and T. D. Millstein. Navigational Plans For Data Integration. In *AAAI/IAAI*, pages 67–73, 1999.

[25] A. Gal, G. A. Modica, H. M. Jamil, and A. Eyal. Automatic Ontology Matching Using Application Semantics. *AI Magazine*, 26(1):21–32, 2005.

[26] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. Intell. Inf. Syst.*, 8(2):117–132, 1997.

[27] 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.

[28] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[29] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pages 1–9, 2006.

[30] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza Peer Data Management System. *IEEE Trans. Knowl. Data Eng.*, 16(7):787–798, 2004.

[31] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *ICDE*, pages 505–516, 2003.

[32] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB J.*, 14(1):68–83, 2005.

[33] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data Integration: The Teenage Years. In *VLDB*, pages 9–16, 2006.

[34] J. Hammer, J. McHugh, and H. Garcia-Molina. Semistructured Data: The Tsimmis Experience. In *ADBIS*, pages 1–8, 1997.

[35] B. He, T. Tao, and K. C.-C. Chang. Organizing structured Web sources by query schemas: a clustering approach. In *CIKM*, pages 22–31, 2004.

[36] H. He, W. Meng, Y. Lu, C. T. Yu, and Z. Wu. Towards Deeper Understanding of the Search Interfaces of the Deep Web. In *World Wide Web*, pages 133–155, 2007.

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

[38] A. Hogue and D. R. Karger. Thresher: automating the unwrapping of semantic content from the World Wide Web. In *WWW*, pages 86–95, 2005.

[39] O. Kaljuvee, O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Efficient Web form entry on PDAs. In *WWW*, pages 663–672, 2001.

[40] J. Kang and J. Choi. Recognising Informative Web Page Blocks Using Visual Segmentation for Efficient Information Extraction. *Journal of Universal Computer Science*, 14(11):1893–1910, 2008.

[41] N. Kushmerick. Learning to Invoke Web Forms. In *CoopIS/DOA/ODBASE*, pages 997–1013, 2003.

[42] A. H. F. Laender, B. A. Ribeiro-Neto, and A. S. da Silva. DEByE - Data Extraction By Example. *Data Knowl. Eng.*, 40(2):121–154, 2002.

[43] J. P. Lage, A. S. da Silva, P. B. Golgher, and A. H. F. Laender. Automatic generation of agents for collecting hidden Web pages for data extraction. *Data Knowl. Eng.*, 49(2):177–196, 2004.

[44] A. Langegger, W. Wöß, and M. Blöchl. A Semantic Web Middleware for Virtual Data Integration on the Web. In *ESWC*, pages 493–507, 2008.

[45] M. Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, 2002.

[46] K. Lerman, L. Getoor, S. Minton, and C. A. Knoblock. Using the Structure of Web Sites for Automatic Segmentation of Tables. In *SIGMOD Conference*, pages 119–130, 2004.

[47] S. W. Liddle, D. W. Embley, D. T. Scott, and S. H. Yau. Extracting Data behind Web Forms. In *ER (Workshops)*, pages 402–413, 2002.

[48] L. Liu, C. Pu, and W. Han. XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources. In *ICDE*, pages 611–621, 2000.

[49] Y. Lu, H. He, H. Zhao, W. Meng, and C. T. Yu. Annotating Structured Data of the Deep Web. In *ICDE*, pages 376–385, 2007.
[50] J. Madhavan, L. Afanasiev, L. Antova, and A. Y. Halevy. Harnessing the Deep Web: Present and Future. In *CIDR*, page To be published, 2009.
[51] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and Reasoning about Mappings between Domain Models. In *AAAI/IAAI*, pages 80–86, 2002.
[52] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-Scale Data Integration: You can afford to Pay as You Go. In *CIDR*, pages 342–350, 2007.
[53] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Y. Halevy. Google's Deep Web crawl. *PVLDB*, 1(2):1241–1252, 2008.
[54] R. McCann, B. K. AlShebli, Q. Le, H. Nguyen, L. Vu, and A. Doan. Mapping Maintenance for Data Integration Systems. In *VLDB*, pages 1018–1030, 2005.
[55] G. A. Modica, A. Gal, and H. M. Jamil. The Use of Machine-Generated Ontologies in Dynamic Information Seeking. In *CoopIS*, pages 433–448, 2001.
[56] P. Montoto, A. Pan, J. Raposo, J. Losada, F. Bellas, and V. Carneiro. A Workflow Language for Web Automation. *Journal of Universal Computer Science*, 14(11):1838–1856, 2008.
[57] P. Montoto, A. Pan, J. Raposo, J. Losada, F. Bellas, and J. López. A Workflow-Based Approach for Creating Complex Web Wrappers. In *WISE*, pages 396–409, 2008.
[58] H. Nguyen, T. Nguyen, and J. Freire. Learning to Extract Form Labels. In *VLDB*, 2008.
[59] A. Pan, P. Montoto, A. Molano, M. Álvarez, J. Raposo, and Á. Viña. A Model for Advanced Query Capability Description in Mediator Systems. In *ICEIS*, pages 140–147, 2002.
[60] M. Petropoulos, A. Deutsch, and Y. Papakonstantinou. Interactive query formulation over Web service-accessed sources. In *SIGMOD Conference*, pages 253–264, 2006.
[61] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2006.
[62] X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2), 2009.
[63] B. Quilitz and U. Leser. Querying Distributed RDF Data Sources with SPARQL. In *ESWC*, pages 524–538, 2008.
[64] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. Technical Report 2000-36, Stanford University, 2000.
[65] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. In *VLDB*, pages 129–138, 2001.
[66] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
[67] C. Rivero. From queries to search forms: an implementation. *IJCAT*, 33(4):264–270, 2008.
[68] A. D. Sarma, X. Dong, and A. Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD Conference*, pages 861–874, 2008.
[69] L. Shu, W. Meng, H. He, and C. T. Yu. Querying Capability Modeling and Construction of Deep Web Sources. In *WISE*, pages 13–25, 2007.
[70] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Comput. Surv.*, 38(2), 2006.
[71] M. Vidal, A. S. da Silva, E. S. de Moura, and J. M. Cavalcanti. Structure-Based Crawling in the Hidden Web. *Journal of Universal Computer Science*, 14(11):1857–1876, 2008.
[72] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *WWW*, pages 187–196, 2003.
[73] M. J. Weal, H. Alani, S. Kim, P. H. Lewis, D. E. Millard, P. A. S. Sinclair, D. D. Roure, and N. R. Shadbolt. Ontologies as facilitators for repurposing web documents. *Int. J. Hum.-Comput. Stud.*, 65(6):537–562, 2007.
[74] Y. Zhai and B. Liu. Structured Data Extraction from the Web Based on Partial Tree Alignment. *IEEE Trans. Knowl. Data Eng.*, 18(12):1614–1628, 2006.
[75] Z. Zhang, B. He, and K. C.-C. Chang. Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax. In *SIGMOD Conference*, pages 107–118, 2004.
[76] Z. Zhang, B. He, and K. C.-C. Chang. Light-weight Domain-based Form Assistant: Querying Web Databases On the Fly. In *VLDB*, pages 97–108, 2005.
[77] H. Zhao, W. Meng, and C. T. Yu. Mining templates from search result records of search engines. In *KDD*, pages 884–893, 2007.