Katrin Braunschweig, Julian Eberius, Maik Thiele, Wolfgang Lehner

Frontiers in Crowdsourced Data Integration

Erstveröffentlichung in / First published in:

Information Technology. 2012, 54 (3), S. 130 – 137 {Zugriff am: 05.11.2020}. De Gruyter. ISSN 2196-7032

DOI: https://doi.org/10.1524/itit.2012.0673

Frontiers in Crowdsourced Data Integration

Probleme und Herausforderungen bei der Crowd-basierten Datenintegration

Katrin Braunschweig, Julian Eberius, Maik Thiele, Wolfgang Lehner, Technische Universität Dresden

Summary There is an ever-increasing amount and variety of open web data available that is insufficiently examined or not considered at all in decision making processes. This is because of the lack of end-user friendly tools that help to reuse this public data and to create knowledge out of it. Therefore, we propose a schema-optional data repository that provides the flexibility necessary to store and gradually integrate heterogeneous web data. Based on this repository, we propose a semi-automatic schema enrichment approach that efficiently augments the data in a "pay-as-you-go" fashion. Due to the inherently appearing ambiguities we further propose a crowd-based verification component that is able to resolve such conflicts in a scalable manner.

►►► **Zusammenfassung** Die stetig wachsende Zahl offen verfügbarer Webdaten findet momentan viel zu wenig oder gar keine Berücksichtigung in Entscheidungsprozessen. Der Grund hierfür ist insbesondere in der mangelnden Unterstützung durch anwenderfreundliche Werkzeuge zu finden, die diese Daten nutzbar machen und Wissen daraus genieren können. Zu diesem Zweck schlagen wir ein schemaoptionales Datenrepositorium vor, welches ermöglicht, heterogene Webdaten zu speichern sowie kontinuierlich zu integrieren und mit Schemainformation anzureichern. Auf Grund der dabei inhärent auftretenden Mehrdeutigkeiten, soll dieser Prozess zusätzlich um eine Crowd-basierende Verifikationskomponente unterstützt werden.

Keywords I.7 [Computing Methodologies: Document and Text Processing]; H.1.2 [Information Systems: Models and Principles: User/Machine Systems]; H.2 [Information Systems: Database Management]; H.3 [Information Systems: Information Storage and Retrieval] ►►► **Schlagwörter** Datenintegration, Crowdsourcing, Kollaboration, WWW

1 Introduction

The World Wide Web is a seemingly unlimited source for data. As shown in Fig. 1, web data can be, for example, the content of a web page or communication snippets from Twitter, blogs or forums. This data often reflects the current state of a society or documents current events. Another type of web data, that is less accessible, is the so-called deep web. It includes all the data found in databases that work as a backend for web forms. The extent of the deep web is difficult to estimate, since it is usually hidden from the average user. Files and documents that can be downloaded from the web are also not directly displayed on the web, but are accessible. The amount of data provided as files has grown significantly as a result of the Open Data trend, which sees government agencies and other organizations opening up their data by providing free download of their files on dedicated public platforms. Furthermore there is a large amount of so-called crowdsourced data from services such as OpenStreetMap. Finally, a relatively new, but constantly growing source of web data is social or collaborative tagging. It allows end-users to provide additional information about the content or meaning of a resource (e. g., web sites, documents, images) using single words or phrases.

Not only does web data come in different shapes and sizes, but it also covers an extremely wide range of domains. Bringing together all these information provides new opportunities for applications. In order to extract the information from the data, we must be able to access and process it. However, accessing the data is difficult, since there is no such thing as a global schema of web data. The goal of web data integration is to provide this uniform

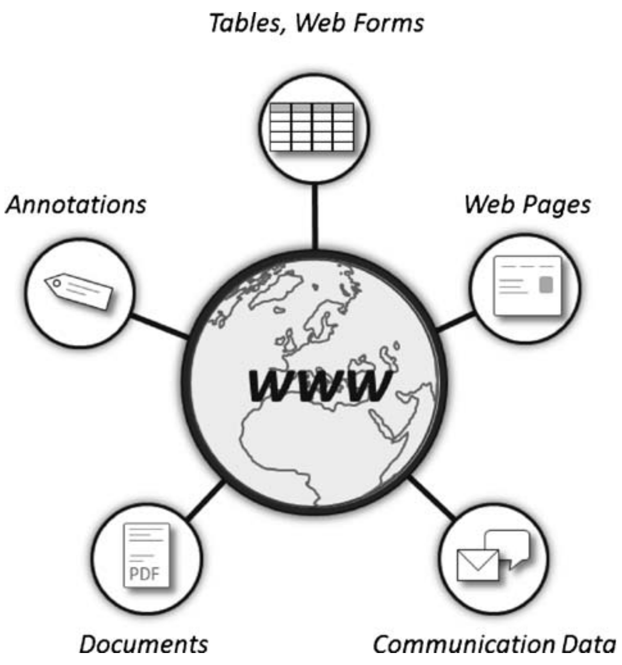

Figure 1 The diversity of web data.

access to the data, regardless of the domain or data type. So far, there is a lack of end-user friendly tools that provide this access for a large, domain-independent scenario as the web. One of the main reasons for that is the fact that traditional data integration solutions developed for the enterprise context are not applicable here. In contrast to the enterprise context, data on the web is not provided or published by the same person or department, which means there are no regulations such as design guidelines or quality control. Furthermore, the data is not limited to a single domain but covers basically every topic humans can think of. This results in a much higher level of diversity and heterogeneity in web data compared to enterprise data. Apart from the data, there are also differences regarding the applications. While enterprise applications tend to be complex and long-running, applications on the web are lightweight and transient. However, traditional tools do not adapt to frequent change that easily. Finally, the user of web applications differs from the traditional enterprise application user. Users on the web are neither domain nor data integration experts. Though some users might have expert knowledge in the field, this assumption does not hold for all users. Therefore, to enable a large number of users to leverage web data, the integration process needs to be simplified. These different characteristics require new approaches for web data integration. Instead of relying on traditional tools, we should focus on new methods that reflect the characteristics of the web.

2 The Challenges of Web Data Integration

Significant differences between data integration in the context of the web and data integration in the enterprise environment prevent the straightforward application of traditional tools. Aspects of the integration process that are similar in both environments, however, may also require changes to the traditional tools due to the much larger scale of the problem on the web. Differences in characteristics and scale can be found in the data itself, in the applications as well as the user.

Data

The most challenging features of web data are without a doubt its diversity and heterogeneity. In contrast to the enterprise environment, where applications usually address only a single domain and domain-specific solutions are beneficial, a data integration system for the web should be generic enough to support all domains. Similar to very large enterprise applications, data volumes on the web can be an issue regarding storage capacity and processing efficiency. Parallel processing and distribution of data across multiple servers is common on the web. The greater challenge for data integration systems, however, is the heterogeneity of the data. Providing a *single view* on the data is crucial for applications, but also very difficult to achieve as it requires domain and technical expertise. Heterogeneity can be found on a structural and a semantic level both in the schema as well as the data.

On the web, we can identify three main types of data. First, there is unstructured data such as text, which contains no schema information. Second, there is regularly structured data, such as relational tables or spreadsheets, where all data instances have the same structure. Semantic heterogeneity may occur between two structured data sets, for example, when the same attribute name is used to describe different concepts. These differences need to be identified to avoid wrong join results. And finally, the third and most prominent category on the web is semi-structured data. Here individual data instances are structured, but not all data instances have the same structure. For example, some instances may have attributes *name* and *size*. Others may also have *name*, but instead of *size* have an attribute *quantity*. Common examples of semi-structured data on the web are all XML-like formats such as HTML web pages or simple annotations. Data formats used for exchanging data between different services on the web are also mostly semi-structured. An irregular structure often results in higher levels of semantic heterogeneity since the schema is inherently more flexible.

Apart from the structural disparity of data sources, we can observe further inconsistencies when examining individual data values. Again, we can see both, semantic as well as structural heterogeneity. Similar to the schema level, semantic heterogeneity on data level includes two, to some extend contrary, concepts. On the one hand, we can find different expressions describing the same concept or thing. For example, *home* and *house* can describe the same entity. The same goes for *river* and *stream*. On the other hand, we can find a long list of words that have more than one meaning. The spelling and even the pronunciation may stay the same, but in different contexts, the word sense changes. For instance, a *ball* can denote both, a piece of sports equipment as well as a festive reception, depending on the context. It is clear that, in both cases, we need to take into account not only the data value itself, but also the context it was published in, to find the correct semantic meaning. Semantic diversity is mainly caused by the fact that the data is not provided by a single authority, but by a large number of people. Since there is no standard way of expressing or describing certain ideas and facts, different people may use different words.

Structural or syntactic heterogeneity in the data is often the result of no standardization. Especially concepts such as names, dates, contact details such as phone numbers, or monetary values feature syntactic differences, since there is more than one way to express them. For example, in some cases a date could be the combination of *day*, *month* and *year*, while in other cases the order might change to *month*, *day*, *year*. These differences pose a great challenge especially for the automatic processing of the data and often require the application of additional cleaning techniques.

Although these characteristics occur both, in the enterprise environment as well as on the web, the absence of control mechanisms on the web means that the level of heterogeneity in the data is usually much higher on the web. Additionally, the wide range of domains further complicates the tasks of semantic disambiguation.

Applications

In the enterprise context, applications that require the integration of data from several resources are typically data warehouse and business intelligence applications. These applications are designed to repeatedly process a number of complex, long-running analysis tasks. The resources are usually a static set of data sources from the same enterprise, which allows developers making certain assumptions about data structure, size and quality. There are no frequent changes to a data warehouse, once the system is running. In contrast, applications on the web are of a more transient nature. User requests change frequently, which means the system must be flexible enough to handle new queries on-demand. Furthermore, the web is not a confined environment like an enterprise. The number of available resources is much higher and applications often access data from multiple unrelated sources. Unfortunately, this means that assumptions about data structure and quality are unlikely to hold across multiple sources. In [3], Franklin et al. address this issue by introducing a new level of abstraction called *dataspaces*. Instead of managing data at the level of individual database systems, a dataspace combines several data sources through a set of relationships. Any kind of relationship, whether it is a simple dependency or a full schema mapping, is recorded in the system and supports the integration process. Another related approach, which addresses the identification and representation of connections between individual data sets is *Linked Data* [1]. Data values from structured data sources are stored as RDF triples consisting of a *subject*, a *predicate* and an *object*. RDF triples are further used to connect data sources, using the subject and object to denote the data objects that are related and the predicate to describe the type of relationship.

In addition to the number of available resources, the range and diversity of the sources is another challenge for applications. Integrating the large number of different domains into a single mediated schema would result in a rigid and unmanageable system. A collection of smaller schemas for individual domains would be more useful. But due to the structural diversity of resources, even matching smaller schemas is very challenging. Different resources may use different schemas to describe the same real-world object. Forcing the data into a single mediated schema would automatically add imprecision to subsequent processing steps. Madhavan et al. [7] address this issue in the context of dataspaces. Instead of enforcing mediated schemata, similar schemas are grouped in clusters, which form the basis for further processing steps. In the same architecture, named *PayGo*, they also use an incremental approach to integration. To address the frequent change in user and application requirements, new relationships between data sources are only added when necessary.

Users

In addition to the data itself and the applications, the user is the third major category that differs considerably on the web, compared to the enterprise environment. The average web user is not a domain expert. Even though individual users do have advanced knowledge of certain domains, it is a difficult task to estimate the skills of the average user. Web users are not a homogenous crowd, still data integration systems should be designed to aid all users. The process of data integration is very hard to automate and human assistance is still required. The scale of the integration task on the web, however, cannot be handled by individual data integration experts. Therefore, classic approaches that can be managed by integration experts are not suitable in this case. Compared to the enterprise environment, the number of so-called end-users, however, is often much larger on the web. Hence, one approach to address this management issue is to shift the costs from individual experts to the large group of users, allowing them to be collaborators instead of just consumers. Web 2.0 technologies enable this form of mass collaboration called *crowdsourcing*.

3 Crowdsourcing

In the web context, crowdsourcing or mass collaboration is a phenomenon with great potential. The web has produced a great number of projects and applications that are the result of the collaborative effort of a large number of contributors, i. e., the crowd. Prominent examples of crowdsourced applications are Wikipedia [16] or OpenStreetMap [15]. A large number of individuals, motivated by a common interest in the topic, contribute their knowledge and/or workforce to reach a common goal, whether that goal is a free digital encyclopedia or a digital world map. There are a number of reasons which motivate a crowdsourcing approach to a problem or task. Of these reasons the following three are highly relevant in the context of web data integration and thus show the potential of crowdsourcing in addressing the issues related to it: First of all, crowdsourcing can reduce both, time and effort, to solve a task by following the classic divide-and-conquer approach. By splitting the task into smaller subtasks and distributing them among several people, the workload is shifted from one person to the crowd. Data integration often requires expert knowledge. However, the scale of web data integration means too much effort for individual integration experts. Therefore, shifting the effort to the crowd is an important step to enable data integration on this large scale. Second, distributing the same subtask to several people can improve precision through redundancy. Since the crowd does not necessarily have expert knowledge, answers of individual people are associated with a level of uncertainty or imprecision. However, having several people solve a task or answer a question and combining these answers, e. g., through majority vote, can significantly reduce the level of imprecision. Finally, an aspect that distinguishes crowdsourcing from other approaches is the fact that tasks are routed to humans to be solved. Especially if certain tasks are too complex or difficult for a computer to solve and would be significantly easier for a human to answer, these tasks can be transferred to humans. A prominent example is image classification. While a computer program can easily scan the individual pixel values of an image, it is very hard for an algorithm to tell what is depicted in the image. That task, however, is straightforward for humans, who can identify the content of an image based on experience. Similar tasks, which require context knowledge and experience, are also part of the data integration process. For example, semantic disambiguation and duplicate detection are tasks that are still very difficult for computer programs to handle, but are integral to high quality data integration. These three aspects presented here show, that crowdsourcing can be very useful to support and enable web data integration. However, the introduction of the crowdsourcing approach into the data integration process also introduces new challenges. These include identifying suitable tasks or question and preparing a strategy how to present these questions to the user. Additionally, a qualified user group needs to be selected and a decision needs to be made whether to only distribute a question or task to a single user or to employ a voting scheme for higher precision. Depending on the selected user group, further attention should be paid to motivating the user to perform the task or give a correct answer and to evaluating the reliability of the user and the correctness of his answer. Some of these challenges have been addressed by researchers. In [4], Franklin et al. present *CrowdDB*, a relational query processing system which incorporates crowdsourcing. Human feedback is requested to either enhance incomplete data or to compare data elements. To enable user feedback, human-oriented query operators are added to the system, which automatically generate user interfaces and present them to the crowd using the microtask crowdsourcing platform *Amazon Mechanical Turk* [14]. This means that not the users of CrowdDB are asked to solve the tasks, but an independent group of users of Mechanical Turk.

Jeffery et al. [6] address the importance of the order, in which tasks are presented to the crowd, in the context of dataspace systems. The overall goal is finding semantic equivalences between data elements from different sources that describe the same real-world concept. First, automatic schema matching and duplicate detection tools are employed to generate so-called candidate matches. The task given to the users is then to either confirm or discard these matches. In the case of mutually exclusive candidate matches, the order in which these matches are presented to the crowd has an impact on the efficiency of the overall algorithm, since confirming one match could automatically discard several other matches. The authors address this challenge using the concept of *value of perfect information*, which can be used to estimate the benefit to the system in case the crowd confirms a match. Matches with the highest benefit are the first to be given to the crowd.

These examples show that crowdsourcing can successfully be applied to individual aspects of the data integration process. However, the benefits of user feedback come at a cost, since crowdsourcing also adds new complex challenges to the process.

4 Connecting the Dots – the Big Picture

The context of the web introduces a number of new challenges to the data integration process. In order to enable applications to efficiently leverage the available data, an integration system for the web needs to address these challenges as a whole, instead of addressing only individual aspects. Figure 2 sketches such a complex approach. As in dataspaces, applications are expected to require data from multiple resources, which may have very different structures. Hence, the integration system should be able to handle structured as well as unstructured data.

Before integrating the data sets, the first important step is to clean the data. The lack of quality control on the web can result in low quality data sets containing many typographical errors and unsuitable formatting. To ensure a certain level of consistency for further processing, data cleaning needs to be applied to all data sets.

4.1 Flexible Data Repository

The structural diversity of web data requires new approaches to overcome the limitations of traditionally

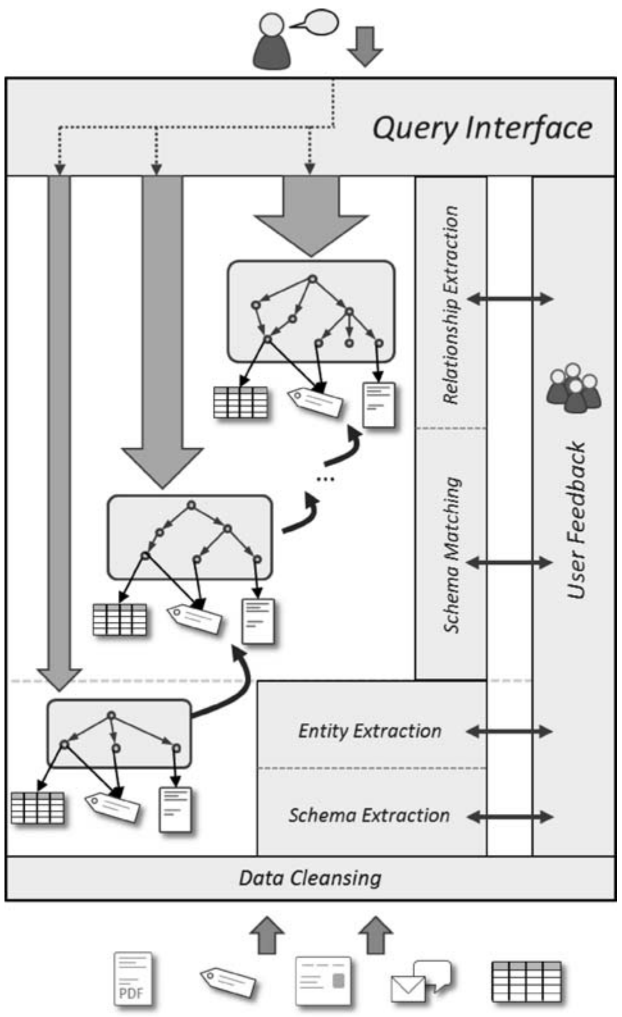

Figure 2 The web data integration process.

databases with their strict schemas that are centrally controlled by a small set of administrators. Two recent and prominent projects in this direction are Freebase [8; 9] and Fluidinfo [17]. Freebase provides a collaboratively created graph database intended to be a public repository of the "world's knowledge". In a similar sense Fluidinfo provides a universal metadata engine. Both projects allow a community-based data integration and both follow the philosophy of "Complete Normalization", i. e., entities are explicitly canonicalized, so one ID represents exactly one real world entity, topic, or concept. These characteristics make them ideal user-friendly platforms for integrating web data where one and the same concept could be represented in many different ways.

In a similar manner we propose a flexible, graph-based data repository, which supports the data integration process by storing schema information, semantic relationships or join options, as the key feature of the integration system. A graph data structure would provide the flexibility required to support different types of meta data, like structural or semantic annotations, and enable complex analysis tasks.

The integration process includes two major steps. First, information about the content and structure of individual data sets needs to be extracted. To extract content information, we can apply entity extraction techniques commonly used for information retrieval from unstructured or semi-structured sources. For structured data, we can additionally apply schema extraction techniques to gather structural information. After collecting the metadata, the second step would be to identify connections between the data sets. These connections include, for example, semantic relationships between entities. If two entities are identified as identical, duplicate entries need to be removed from the system. To find connections between schemas extracted from structured data, various schema matching as well as mapping techniques can be applied.

Indexing the data using the extracted information ultimately enables the search for concepts across multiple sources and the identification of join options.

As depicted in Fig. 2, the integration process should not be performed once for the entire data set collection, but in smaller, incremental steps. At first, data sets can be collected in a staging area without being processed immediately. The integration process is then triggered by user and application requirements and would be performed to an extend that meets these requirements. This approach could save costs for integration effort that is not necessarily required for the applications. Once the effort has been made to integrate data sets, the results can be stored in our data repository and reused for subsequent requests. This incremental approach is similar to the *pay-as-you-go* fashion proposed in [7].

4.2 Getting the User Involved

As mentioned before, a complete automation of the integration process is very difficult and highly unlikely. As Alon Halevy points out in [5]: "*Resolving schema heterogeneity is inherently a heuristic, human assisted process.*" The same can be said for entity resolution as well as semantic disambiguation. For several subtasks in the integration process, automatic tools can only provide approximate results. That means that we need human feedback, in order to achieve accurate results. Due to the scale of the task, it is unrealistic from an economical point of view to employ integration expert. To avoid an expert-in-the-loop approach, our suggestion is to get the huge crowd of non-expert users involved to shift the costs. In order to make the integration tasks easy enough to be solved by non-expert users we need to apply semi-automatic extraction and matching tools generating potential candidates which are later verified by the crowd-workers. The whole entity and relationship extraction as well as schema enrichment process is therefore augmented with an additional crowd component (see Fig. 2) that contributes directly to the integration process.

Another important part within the interaction with the crowd-workers is the careful design of the crowd workflows and user interfaces to achieve a high answer quality as well as low latencies. Tasks that are too complex for a single working step must be detected and decomposed in smaller subtasks. Bernstein et al. [10] therefore proposes a crowd programming pattern called Find-Fix-Verify that decomposes complex tasks into a sequence of generation and review stages that utilize independent agreement and voting to produce high-quality results. Despite the overall process design each individual crowd-task needs precise instructions that clearly define the expected outcome.

Obviously such a crowd-assisted data integration approach heavily relies on crowd-workers that are willing to contribute their manpower. Whereas many crowd-workers are motivated by the micro-payments on platforms such as Amazon Mechanical Turk or Crowdflower, there is also a significant amount of workers with other motivational factors. A previous study with the title "*More than fun and money. Worker Motivation in Crowdsourcing*" [11] classifies the motivating factors in *Enjoyment Based Motivation* and *Community Based Motivation* (intrinsic types) as well as *Immediate Payoffs*, *Delayed Payoffs* and *Social Motivation* (extrinsic types). The result of the study showed, that beside the immediate payoffs also the enjoyment factor played an import role in user motivation. Accordingly, it can be concluded that the crowd-tasks need to be carefully designed in order to provide an enjoyable experience. A good example in this sense are the so-called games with a purpose [12].

4.3 Query Interface

To enable web users and applications to leverage the data, a potential system must provide a consistent interface for on-demand queries across multiple data sources. With the integration as an incremental approach, different data sets can be in different stages of the process. Still, basic keyword search, as it is common on the web, should be available even at the lower stages. Progress in the integration process should automatically enable more complex queries required for applications. The increasing range and complexity of queries is also depicted in Fig. 2. We argue that providing a uniform query interface and incorporating interactive query refinement, if the underlying structure allows for more complex queries, could assist in hiding the structural heterogeneity of the different stages from the user. It would create a more intuitive query process for non-expert users.

5 Conclusion

Significant differences between the enterprise environment and the web represent new challenges for the integration of heterogeneous data sources into a consistent model. Traditional tool are not designed to handle these challenges and are often not applicable in this new environment. To address this issue, there is substantial ongoing research in the field of web data integration. This includes both, developing new approaches which directly address the characteristics of the web, as well as adjusting traditional approaches to meet the new requirements. Significant contributions are the concept of dataspaces as a new abstraction level for data integration and management and the application of crowdsourcing further support.

References

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. In: *Int'l Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[2] L. Chiticariu, M. A. Hern‡ndez, P. G. Kolaitis, and L. Popa. Semi-Automatic Schema Integration in Clio. In: *Proc. of the 33rd Int'l Conf. on Very large Data Bases (VLDB)*, pages 1326–1329, 2007.

[3] M. Franklin, A. Halevy and D. Maier. From Databases to Dataspaces: A New Abstraction for Information Management. In: *ACM SIGMOD Record*, 34(4):27–33, 2005.

[4] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: Answering Queries with Crowdsourcing. In: *Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data*, pages 61–72, 2011.

[5] A. Halevy. Why Your Data Won't Mix. In: *Queue*, 3(8):50–58, 2005.

[6] S. R. Jeffery, M. J. Franklin, and A Y. Halevy. Pay-as-you-go User Feedback for Dataspace Systems. In: *Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data*, pages 847–860, 2008.

[7] J. Madhavan, S. R. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, and A. Y. Halevy. Web-scale Data Integration: You can only afford to Pay as you Go. In: *Proc. of the 3rd Biennial Conf. on Innovative Data Systems Research (CIDR)*, pages 164–168, 2007.

[8] K. Bollacker, R. Cook, and P. Tufts. Freebase: a shared database of structured general human knowledge. In: *Proc. of the 22nd National Conf. on Artificial Intelligence (AAAI)*, Volume 2, pages 1962–1963, 2007.

[9] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In: *Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data*, pages 1247–1250, 2008.

[10] M. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: A Word Processor with a Crowd Inside. In: *Proc. of the 23nd Annual ACM Symp. on User Interface Software and Technology (UIST)*, pages 313–322, 2010.

[11] N. Kaufmann, T. Schulze, and D. Veit. More than fun and money. Worker Motivation in Crowdsourcing – A Study on Mechanical Turk. In: *Proc. of the 17th Americas Conf on Information Systems (AMCIS)*, 2011.

[12] L. von Ahn and L. Dabbish. Designing games with a purpose. In: *Communications of the ACM*, 51(8):58–67, 2008.

[13] CrowdFlower. www.crowdflower.com/

[14] Amazon Mechanical Turk. www.mturk.com

[15] OpenStreetMap. www.openstreetmap.org

[16] Wikipedia – The Free Enzyclopedia. www.wikipedia.org

[17] fluidinfo. www.fluidinfo.com

Received: November 6, 2011, accepted: March 22, 2012

Katrin Braunschweig is a PhD student and member of the junior research group EDYRA. She received her Media Computer Science master's degree from the Technische Universität Dresden in May 2010. In her masters's thesis she studied the suitability of graphics processing units for the task of uncertain data mining.

Address: Database Technology Group, Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany, Tel.: +49-351-46338402, e-mail: katrin.braunschweig@tu-dresden.de

Julian Eberius is a Ph.D. student and member of the junior research group EDYRA. He received his Media Computer Science master's degree from the Technische Universität Dresden in December 2010. For his thesis he developed a method for automatically configuring adaptive schema matching systems. From 2008 to 2010, he was a student research assistant at SAP Research Dresden.

Address: Database Technology Group, Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany, Tel.: +49-351-46338402, e-mail: julian.eberius@tu-dresden.de

Dr.-Ing. Maik Thiele finished his dissertation on "Quality-Driven Data Production Controlling in Real-Time DW Systems" in May 2010 and received his doctorate with distinction. His research interests include real-time data warehousing and situational data analysis. He also contributes to several industrial projects in cooperation with GfK Nuremberg or UBS Zurich.

Address: Database Technology Group, Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany, Tel.: +49-351-46338283, e-mail: maik.thiele@tu-dresden.de

Prof. Dr.-Ing. Wolfgang Lehner received his Master, Ph.D., and habilitation in Computer Science from the University of Erlangen-Nuremberg. Since 2002, Wolfgang Lehner is full professor and head of the Database Technology Group at TU Dresden. He was a visiting scientist at IBM Almaden, Microsoft Research Redmond, and SAP Walldorf. His major research focuses on the efficient processing of empirically collected mass data with advanced database technology.

Address: Database Technology Group, Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany, Tel.: +49-351-46338383, e-mail: wolfgang.lehner@tu-dresden.de