Data Profiling Revisited

Felix Naumann*
Qatar Computing Research Institute (QCRI), Doha, Qatar
fnaumann@qf.org.qa

ABSTRACT

Data profiling comprises a broad range of methods to efficiently analyze a given data set. In a typical scenario, which mirrors the capabilities of commercial data profiling tools, tables of a relational database are scanned to derive metadata, such as data types and value patterns, completeness and uniqueness of columns, keys and foreign keys, and occasionally functional dependencies and association rules. Individual research projects have proposed several additional profiling tasks, such as the discovery of inclusion dependencies or conditional functional dependencies.

Data profiling deserves a fresh look for two reasons: First, the area itself is neither established nor defined in any principled way, despite significant research activity on individual parts in the past. Second, more and more data beyond the traditional relational databases are being created and beg to be profiled. The article proposes new research directions and challenges, including interactive and incremental profiling and profiling heterogeneous and non-relational data.

1. DATA PROFILING

"*Data profiling is the process of examining the data available in an existing data source [...] and collecting statistics and information about that data.*"[1] Profiling data is an important and frequent activity of any IT professional and researcher. We can safely assume that any reader of this article has engaged in the activity of data profiling, at least by eye-balling spreadsheets, database tables, XML files, etc. Possibly more advanced techniques were used, such as key-word-searching in data sets, sorting, writing structured queries, or even using dedicated data profiling tools. While the importance of data profiling is undoubtedly high, and while efficiently and effectively profiling is an enormously difficult challenge, it has yet to be established as a research area in its own right. We focus our discussion on relational data, the predominant format of traditional data profiling methods, but we do regard data profiling for other data models in a separate section.

Data profiling encompasses a vast array of methods to examine data sets and produce metadata. Among the simpler results are statistics, such as the number of null values and distinct values in a column, its data type, or the most frequent patterns of its values. Metadata that are more difficult to compute usually involve multiple columns, such as inclusion dependencies or functional dependencies. More advanced techniques detect approximate properties or conditional properties of the data set at hand. To allow focus, the broad field of data mining is deliberately omitted from the discussion here, as justified below. Obviously, all such discovered metadata refer only to the given data instance and cannot be used to derive with certainty schematic/semantic properties, such as primary keys or foreign key relationships. Figure 1 shows a classification of data profiling tasks. The tasks for "single sources" correspond to state-of-the-art in tooling and research (see Section 2), while the tasks for "multiple sources" reflect new research directions for data profiling (see Section 5).

Systematic data profiling, i.e., profiling beyond the occasional exploratory SQL query or spreadsheet browsing, is usually performed by dedicated tools or components, such as IBM's Information Analyzer, Microsoft's SQL Server Integration Services (SSIS), or Informatica's Data Explorer. Their approaches all follow the same general procedure: A user specifies the data to be profiled and selects the types of metadata to be generated. Next, the tool computes in batch the metadata using SQL queries and/or specialized algorithms. Depending on the volume of the data and the selected profiling results, this step can last minutes to hours. The results are usually displayed in a vast collec-

*On leave from Hasso Plattner Institute, Potsdam, Germany (naumann@hpi.uni-potsdam.de).
[1]Wikipedia on "Data Profiling", 2/2013

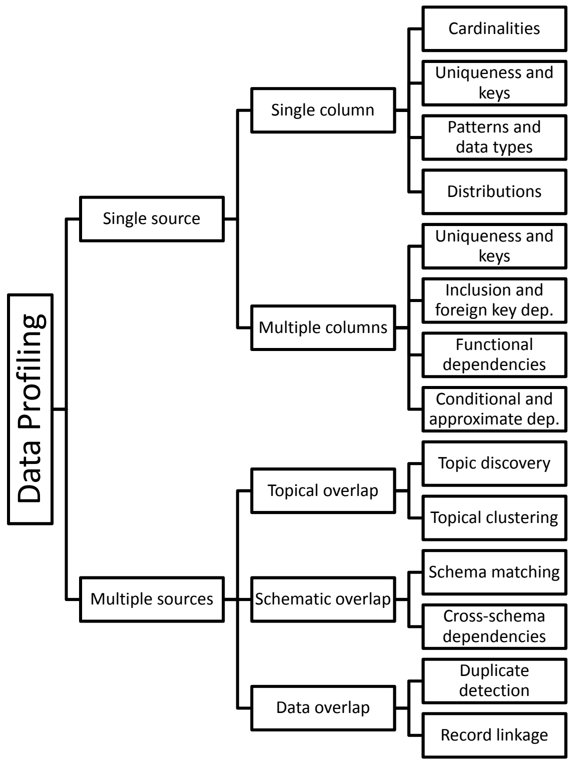

Figure 1: A classification of data profiling tasks

tion of tabs, tables, charts, and other visualizations to be explored by the user. Typically, discoveries can then be translated into constraints or rules that are then enforced in a subsequent cleansing/integration phase. For instance, after discovering that the most frequent pattern for phone numbers is `(ddd)ddd-dddd`, this pattern can be promoted to the *rule* that all phone numbers must be formatted accordingly. Most cleansing tools can then either transform differently formatted numbers or at least mark them as violations.

Use cases for profiling. The need to profile a new or unfamiliar set of data arises in many situations, in general to prepare for some subsequent task.

Query optimization. Basic profiling is performed by most database management systems to support query optimization with statistics about tables and columns. These profiling results can be used to estimate the selectivity of operators and ultimately the cost of a query plan.

Data cleansing. Probably the most typical use case is profiling data to prepare a data cleansing process. Profiling reveals data errors, such as inconsistent formatting within a column, missing values, or outliers. Profiling results can also be used to measure and monitor the general quality of a data set, for instance by determining the number of records that do not conform to previously established constraints.

Data integration. Often the data sets to be integrated are somewhat unfamiliar and the integration expert wants to explore the data sets first: How large is it? What data types are needed? What are the semantics of columns and tables? Are there dependencies between tables and among databases, etc.? The vast abundance of (linked) *open data* and the desire and potential to integrate them with enterprise data has amplified this need.

Scientific data management. The management of data that is gathered during scientific experiments or observations has created additional motivation for efficient and effective data profiling: When importing raw data, e.g., from scientific experiments or extracted from the Web, into a DBMS, it is often necessary and useful to profile the data and then devise an adequate schema.

Data analytics. Almost any statistical analysis or data mining run is preceded by a profiling step to help the analyst understand the data at hand and appropriately configure tools, such as SPSS or Weka. Pyle describes detailed steps of analyzing and subsequently preparing data for data mining [38].

Knowledge about data types, keys, foreign keys, and other constraints supports data modeling and helps keep data consistent, improves query optimization, and reaps all the other benefits of structured data management. Other research efforts have mentioned query formulation and indexing [42], scientific discovery [26], and database reverse engineering [35] as further motivation for data profiling.

Time to revisit. Recent trends in the database field have added challenges but also opportunities for data profiling. First, under the *big data* umbrella, industry and research have turned their attention to data that they do not own or have not made use of yet. Data profiling can help assess which data might be useful and reveals the yet unknown characteristics of such new data: before exposing an infrastructure to Twitter's firehose it might be worthwhile to know about properties of the data one is receiving; before downloading significant parts of the linked data cloud, some prior sense of the integration effort is needed; before augmenting a warehouse with text mining results an understanding of their quality is required. Leading researchers have recently noted "*If we just have a bunch of data sets in a repository, it is unlikely anyone will ever be able to find, let alone reuse, any of this data. With adequate metadata, there is some hope, but even so, challenges will remain* [...]" [4].

Second, much of the data that shall be exploited is of non-traditional type for data profiling, i.e., non-relational (e.g., linked open data), non-structured (e.g., tweets and blogs), and heterogeneous (e.g., open government data). And it is often truly "big", both in terms of schema, rendering algorithms that are exponential in the number of schema elements infeasible, and in terms of data, rendering main-memory based methods infeasible. Existing profiling methods are not adequate to handle that kind of data: Either they do not scale well (e.g., dependency discovery), or there simply are no methods yet (e.g., incremental profiling, profiling multiple data sets, profiling textual attributes).

Third, different and new data management architectures and frameworks have emerged, including distributed systems, key-value stores, multi-core- or main-memory-based servers, column-oriented layouts, streaming input, etc. These new premises provide interesting opportunities as we discuss later.

Profiling challenges. Data profiling, even in a traditional relational setting, is non-trivial for three reasons: First, the results of data profiling are *computationally complex* to discover. For instance, discovering key candidates or dependencies usually involves some sorting step for each considered column. Second, the *discovery-aspect* of the profiling task demands the verification of complex constraints on all columns and combinations of columns in a database. And thus also the solution-space of uniqueness-, inclusion dependency-, or functional dependency-discovery is exponential in the number of attributes. Third, profiling is often performed on data sets that may not fit into main memory.

Various tools and algorithms have tackled these challenges in different ways. First, many rely on the capabilities of an underlying DBMS, as many profiling tasks can be expressed as SQL queries. Second, many have developed innovative ways to handle the individual challenges, for instance using indexing schemes, parallel processing, and reusing intermediate results. Third, several methods have been proposed that deliver only approximate results for various profiling tasks, for instance by profiling samples. Finally, users are asked to narrow down the discovery process to certain columns or tables. For instance, there are tools that verify inclusion dependencies on user-suggested pairs of columns, but that cannot automatically check inclusion between all pairs of columns or column sets.

The following section elaborates these traditional data profiling tasks and gives a brief overview of known approaches. Sections 3 – 6 are the main contributions of this article by defining and motivating new research perspectives for data profiling. These areas include interactive profiling (users can act upon profiling results and re-profile efficiently), incremental profiling (profiling results are incrementally updated as new data arrives), profiling heterogeneous data and multiple sources simultaneously, profiling non-relational data (XML and RDF), and profiling on different architectures (column stores, key-value stores, etc.).

This article is not intended to be a survey of existing approaches, though there is certainly a need for such, nor is it a formal framework for future data profiling developments. Rather, it strives to spark interest in this research area and to assemble a wide range of research challenges.

2. STATE OF THE ART

While the introduction mentions current industrial profiling tools, this section discusses current research directions. In its basic form, data profiling is about analyzing data values of a single column, summarized as "traditional data profiling". More advanced techniques detect relationships among columns of one or more tables, which we discuss as "dependency detection". Finally, we distinguish data profiling from the broad field of "data mining", which we deliberately exclude from further discussion.

Traditional data profiling. The most basic form of data profiling is the analysis of individual columns in a given table. Typically, generated metadata comprises various counts, such as the number of values, the number of unique values, and the number of non-null values. These metadata are often part of the basic statistics gathered by DBMS. Mannino et al. give a much-cited survey on statistics collection and its relationship to database optimization [32]. In addition to the basic counts, the maximum and minimum values are discovered and the data type is derived (usually restricted to string vs. numeric vs. date). Slightly more advanced techniques create histograms of value distributions, for instance to optimize range-queries [37], and identify typical patterns in the data values in the form of regular expressions [40]. Data profiling tools display such results and can suggest some actions, such as declaring a column with only unique values a key-candidate or suggesting to enforce the most frequent patterns.

Dependency detection. Dependencies are metadata that describe relationships among columns. The difficulties are twofold: First, *pairs* of columns or column-sets must be regarded, and second, the chance existence of a dependency in the data at hand does not imply that this dependency is *meaningful*.

The most frequent real-world use-case is the discovery of foreign keys [30, 41] with the help of inclusion dependencies [6, 33]. Current data profiling tools often avoid checking all combinations of columns, but rather ask the user to suggest a candidate key/foreign-key pair to verify. Another form of dependency, which is also relevant for data quality, is the functional dependency (FD). Again, much research has been performed to automatically detect FDs [26, 45].

Both types of dependencies can be relaxed in two ways. First, *conditional dependencies* need to hold only for tuples that fulfill the condition. Conditional inclusion dependencies (CINDs) were proposed for data cleaning and contextual schema matching [11]. Different aspects of CIND discovery have been addressed in [5, 17, 22, 34]. Conditional functional dependencies (CFDs) were introduced in [20] for data cleaning. Algorithms for discovering CFDs are also proposed in [14, 21]. Second, *approximate dependencies* need to hold only for a certain percentage of the data – they are not guaranteed to hold for the entire relation. Such dependencies are often discovered using sampling [27] or other summarization techniques [16].

Finally, algorithms for the discovery of columns and column combinations with only unique values (which is strictly speaking a constraint and not a dependency) have been proposed in [2, 42].

To reiterate our motivation: There are various individual techniques for various individual profiling tasks. What is lacking even for the state-of-the-art is a unified view of data profiling as a field and a unifying framework of its tasks.

Data mining. Rahm and Do distinguish data profiling from data mining by the number of columns that are examined: "Data profiling focusses on the instance analysis of individual attributes. [...] Data mining helps discover specific data patterns in large data sets, e.g., relationships holding between several attributes" [39]. Yet, a different distinction is more useful to separate the different use cases: Data profiling gathers technical metadata to support data management, while data mining and data analytics discovers non-obvious results to support business management. In this way, data profiling results are information about columns and column sets, while data mining results are information about rows or row sets (clustering, summarization, association rules, etc.).

Of course such a distinction is not strict. Some data mining technology does express information about columns, such as feature selection methods for sets of values within a column [7] or regression techniques to characterize columns [13]. Yet with the distinction above, we concentrate on data profiling and put aside the broad area of data mining, which has already received unifying treatment in numerous text books and surveys.

3. INTERACTIVE DATA PROFILING

Data profiling research has yet hardly recognized that data profiling is an inherently user-oriented task. In most cases, the produced metadata is consumed directly by the user or it is at least regarded by a user before put to use in some application, such as schema design or data cleansing. We suggest the involvement of the user already during the algorithmic part of data profiling, hence "interactive profiling".

Online profiling. Despite many optimization efforts, data profiling might last longer than a user is willing to wait in front of a screen with nothing to look at. Online profiling shows intermediate results as they are created. However, simply hooking the graphical interface into existing algorithms is usually not sufficient: Data that is sorted by some attribute or has a skewed order yields misleading intermediate results. Solutions might be approximate or sampling-based methods, whose results gracefully improve as more computation is invested. Naturally, such intermediate results do not reflect the properties of the entire data set. Thus, some form of confidence, along with a progress indicator, can be shown to allow an early interpretation of the results.

Apart from entertaining users during computation, an advantage of online profiling is that the user may abort the profiling run altogether. For instance, a user might decide early on that the data set is not interesting (or clean) enough for the task at hand.

Profiling on queries and views. In many cases, data profiling is performed with the purpose of cleaning the data or the schema to some extent, for instance, to be able to insert it into a data warehouse or to integrate it with some other data set. However, each cleansing step changes the data, and thus implicitly also the metadata produced by profiling. In general, after each cleansing step a new profiling run should be performed. For instance, only after cleaning up zip codes does the functional dependence with the city values become apparent. Or only after deduplication does the uniqueness of email addresses reveal itself.

A modern profiling system should be able to allow users to virtually interact with the data and re-compute profiling results. For instance, the profiling system might show a 96% uniqueness for a certain column. The user might recognize that indeed the attribute should be completely unique and is in fact a key. Without performing the actual cleansing, a user might want to virtually declare the column to be a key and re-perform profiling on this virtually cleansed data. Only then a foreign key for this attribute might be recognized.

In short, a user might want to act upon profiling results in an ad-hoc fashion without going through the entire cleansing and profiling loop, but remain within the profiling tool context and perform cleansing and re-profiling only on a virtually cleansed view. When satisfied, the virtual cleansing can of course be materialized. A key enabling technology for this kind of interaction is the ability to efficiently re-perform profiling on slightly changed data, as discussed in the next section. In the same manner, profiling results can be efficiently achieved on query results: While calculating the query result, profiling results can be generated on the side, thus showing a user not only the result itself, but also the nature of that data. Faceted search provides similar features in that a user is presented with cardinalities based on the chosen filters.

For all suggestions above, new algorithms and data structures are needed to enhance the user experience of data profiling.

4. INCREMENTAL DATA PROFILING

A data set is hardly ever fixed: Transactional data is appended to frequently, analytics-oriented data sets experience periodic updates (typically daily), and large data sets available on the web data are updated every few weeks or months. Data profiling methods should be able to efficiently handle such moving targets, in particular without re-profiling the entire data set.

Incremental profiling. An obvious, but yet under-examined extension to data profiling is to re-use earlier profiling results to speed-up computation on changed data. I.e., the profiling system is provided with a data set and with knowledge of its delta compared to a previous version, and it has stored any intermediate or final profiling results on that previous version. In the simplest cases, profiling metadata can be calculated associatively (e.g., sum, count, equi-width histograms), in some cases some intermediate metadata can help (e.g., sum and count for average, indexes for value patterns), and finally in some cases a complete recalculation might be necessary (e.g., median or clustering).

There is already some research on performing individual profiling tasks incrementally. For instance, the AD-Miner algorithm allows an incremental update of functional dependency information [19]. Fan et al. focus on the area of conditional functional dependencies and also consider incremental updates [20]. The area of data mining, on the other hand, has seen much related work, for instance on association rule mining and other data mining applications [24].

Continuous profiling. While for incremental profiling we assumed periodic updates (or periodic profiling runs), a further use case is to update profiling results while (transactional) data is created or updated. If the profiling results can be expressed as a query, and if they shall be performed only on a temporal window of the data, this use case can be served by data stream management systems [23]. If this is not the case, continuous profiling methods need to be developed, whose results can be displayed in a dashboard. Of particular importance is to find a good tradeoff between recency, accuracy, and resource consumption. Use cases for continuous profiling include internet traffic monitoring or the profiling of incoming search queries.

Multi-measure profiling. Each profiling algorithm has its own scheme of running through the data and collecting or aggregating whatever information is needed. Realizing that multiple types of profiling metadata shall be collected, it is likely that many of these runs can be combined. Thus, in a manner similar to multi-query-optimization, there is a high potential for efficiency gains, in particular wrt. I/O cost. While such potential is already realized in commercial systems, it has not yet been investigated for the more complex tasks that are not covered by these tools.

5. PROFILING HETEROGENEOUS DATA

While typical profiling tasks assume a single, largely homogeneous database or even only a single table, there are many use cases in which a combined profiling of multiple, heterogeneous data sets is needed. In particular when integrating data it is useful to learn about the *common properties* of participating data sets. From profiling one can learn about their integrability, i.e., how well their data and schemata fit together, and learn in advance the properties of the integrated data set. Even profiling a single source that stores data for multiple or many domains, such as DBpedia or Freebase, can profit from techniques that profile heterogeneous data.

Degrees of heterogeneity. Heterogeneity in data sets can appear at many different levels and in many different degrees of severity. Data profiling methods can be used to uncover these heterogeneities and possibly provide hints on how to overcome them.

Heterogeneity is traditionally divided into syntactic heterogeneity, structural heterogeneity, and semantic heterogeneity [36]. Discovering syntactic heterogeneity, in the context of data profiling, is precisely what traditional profiling aims at, e.g., finding inconsistent formatting. Next, structural heterogeneity appears in the form of unmatched schemata and differently structured information. Such problems are only partly addressed by traditional profiling, e.g., by discovery schema information, such as types, keys, or foreign keys. Finally, semantic heterogeneity addresses the underlying and possibly mismatched meaning of the data. For data profiling we interpret it as the discovery of semantical overlap of the data and their domain(s).

Data profiling for integration. Our focus here is on profiling tasks to discover structural and semantic heterogeneity, arguing that structural profiling seeks information about the schema and semantic profiling seeks information about the data. Both serve to assess the *integrability* of data sets, and thus also indicate the necessary integration effort, which is vital to project planning. The integration effort might be expressed in terms of similarity, but also in terms of man-months or in terms of which tools are needed.

An important issue in integrated information systems, irrelevant for single databases, is the *schematic similarity*, i.e., the degree to which their schemata complement each other and the degree to which they overlap. There is an obvious relation to schema matching techniques, which aim at automatically finding correspondences between schema elements [18]. Already Smith et al. have recognized that schema matching techniques often play the role of profiling tools [43]: Rather than using them to derive schema mappings and perform data transformation, they play roles that have a more informative character, such as assessment of project feasibility or the identification of integration targets. However, the mere matching of schema elements might not suffice as a profiling-for-integration result: Additional information on the structure of the values of the matching columns can provide further details about the integration difficulty.

After determining schematic overlap, a next step is to determine *data overlap*, i.e., the (estimated) number of real-world objects that are represented in both data sets, or that are represented multiple times in a single data set. Such multiple representations are typically identified using entity matching methods (aka. record linkage, entity resolution, duplicate detection, and many other names) [15]. However, estimating the number of matches without actually performing the matching on the entire data set is an open problem. If used to determine the integration effort, it is additionally important to know how diverse such matching records are represented, i.e., how difficult it is to devise good similarity measures and find appropriate thresholds.

Topical profiling. When profiling yet unknown data from a large pool of sources, it is necessary to recognize the topic or domain covered by the source. One recently proposed use case for such source discovery is situational BI where warehouse data is complemented with data from openly available sources [3, 31]. Examples for such sources are the set of linked open data sources (`linkeddata.org`) or tables gleaned from the web: "Data on the Web reflects every topic in existence, and topic boundaries are not always clear." [12]

Topical profiling should be able to match a data set to a given set of topics or domains. Given two data sets, it should be able to determine topical overlap between them. There is already initial work on topical profiling for traditional databases in the iDisc system [44], which matches tables to topics or clusters them by topic, and for web data [8], which discovers frequent patterns of concepts and aggregates them to topics.

6. DATA PROFILING ON OTHER ARCHITECTURES

Most current data profiling methods and tools assume data to be stored in relational form on a single-node database. However, much interesting data nowadays resides in data stores of different architecture and in various (non-relational) models and formats. If these architectures are more amenable to data profiling tasks, they might even warrant copying data for the purpose of profiling.

Storage architectures. Of all modern hardware architectures, columnar storage seems the most promising for many data profiling tasks, which often are inherently column-oriented: Analyzing individual columns for patterns, data types, uniqueness, etc. involves reading only the data of that column and thus matches precisely the sweet-spot of columns stores [1]. This advantage may dwindle when analyzing column-combinations, for instance to discover functional dependencies, but even then one can avoid reading entire rows of data.

As data profiling includes many different tasks on many tables and columns, a promising research avenue is the use of many cores, GPUs, or distributed environments for parallelization. Parallelization can occur at different levels: A comprehensive profiling run might distribute individual, independent profiling tasks to different nodes (task parallelism). Another approach is to partition data for a single profiling task (data parallelism). As most profiling tasks are not associative, in the sense that profiling results for subsets of column-values cannot be aggregated to overall results, horizontal partitioning is usually not useful or at least raises some coordination overhead. For instance, uniqueness within each partition of a column does not imply uniqueness of the entire column, but communicating the sets of distinct values is sufficient. Finally, task parallelism can again be applied to finer-grained tasks, such as sorting or hashing, that form the basic building blocks of many profiling algorithms.

Further challenges arise when performing data profiling on key-value stores: Typically, the values contain some structured data, without enforced schemata. Thus, even defining the expected results on such “soft schema” values is a challenge, and a first step must involve schema profiling as described in Section 5.

To systematically evaluate different methods and architectures for the various data profiling tasks, a corresponding *data profiling benchmark* is needed. It must define (i) a set of tasks, (ii) data on which the tasks shall be executed, and (iii) measures to evaluate efficiency. For (i) the first (single-source) subtree of Figure 1 can serve as an initial set of tasks. Arguably, the most difficult part of establishing a benchmark is to (ii) provide data that closely mirrors real-world situations. Given a schema and a set of constraints (uniqueness, data types, FDs, INDs, patterns, etc.) it is not trivial to create a valid database instance. If in addition some dirtiness, i.e., violations to constraints, are to be inserted, or if conditional dependencies are needed, the task becomes even more daunting. The measures for (iii) need to be carefully selected, in particular if they are to go beyond traditional measures of response time and cost efficiency and include the evaluation of approximate results. Finally, the benchmark should be able to evaluate not only entire profiling systems but also methods for individual tasks.

Types of data. Data comes not only in relational form, but also in tree or graph shapes, such as XML and RDF data. A first step is to adapt traditional profiling tasks to those models. An example is ProLOD, which profiles linked open data delivered as RDF triples [10]. A further challenge arises from the sheer size of many RDF data sets, so profiling computation must be distributed [9]. In addition, such data models demand new, data model-specific profiling tasks, such as maximum tree depth or average node-degree.

Structured data is often intermingled with unstructured, textual data, for instance in product information or user profiles on the web. The field of linguistics knows various measures to characterize a text from simple measures, such as average sentence length, to complex measures, such as vocabulary richness [25] as visualized in [29]. Thus, data profiling might be extended to text profiling and possibly to methods that jointly profile both data and text. A discussion on the large area of text mining is omitted, for the same reasons data mining was omitted from this article.

7. AN OUTLOOK

This article points out the potentials and the needs of modern data profiling – there is yet much principled research to do. A planned first step is to develop a general framework for data profiling, which classifies and formalizes profiling tasks, shows its amenability for a range of use cases, and provides a means to compare various techniques both in their abilities and their efficiency.

At the same time, this article shall serve as a “call to arms” for database researchers to develop more efficient and more advanced profiling techniques, in particular for the fast growing areas of “big data” and “linked data”, both of which have attracted great interest by industry, but both of which have proven that data is difficult to grasp and use effectively. Data profiling can bridge this gap by showing what the data sets are about, how well they fit the data environment at hand, and what steps are needed to make use of them.

Several research areas were deliberately omitted in this article, in particular data mining and text mining, as reasoned above, but also data visualization: Because data profiling targets users, effectively visualizing the profiling results is of utmost importance. A suggestion for such a visual data profiling tool is the Profiler system [28]. A strong cooperation between the database community, which produces the data and metadata to be visualized, and the visualization community, which enables users to understand and make use of the data, is needed.

Acknowledgments. Discussions and collaboration with Ziawasch Abedjan, Jana Bauckmann, Christoph Böhm, and Frank Kaufer inspired this article.

8. REFERENCES

[1] D. J. Abadi. Column stores for wide and sparse data. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, pages 292–297, Asilomar, CA, 2007.

[2] Z. Abedjan and F. Naumann. Advancing the discovery of unique column combinations. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1565–1570, Glasgow, UK, 2011.

[3] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazón, F. Naumann, T. B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen. Fusion Cubes: Towards self-service business intelligence. *Data Warehousing and Mining (IJDWM)*, in press, 2013.

[4] D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, U. Dayal, M. Franklin, J. Gehrke, L. Haas, A. Halevy, J. Han, H. V. Jagadish, A. Labrinidis, S. Madden, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, K. Ross, C. Shahabi, D. Suciu, S. Vaithyanathan, and J. Widom. Challenges and opportunities with Big Data. Technical report, Computing Community Consortium, `http://cra.org/ccc/docs/init/bigdatawhitepaper.pdf`, 2012.

[5] J. Bauckmann, Z. Abedjan, H. Müller, U. Leser, and F. Naumann. Discovering conditional inclusion dependencies. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2094–2098, Maui, HI, 2012.

[6] J. Bauckmann, U. Leser, F. Naumann, and V. Tietz. Efficiently detecting inclusion dependencies. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 1448–1450, Istanbul, Turkey, 2007.

[7] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the Conference on Advanced Information Systems Engineering (CAiSE)*, pages 452–466, Toronto, Canada, 2002.

[8] C. Böhm, G. Kasneci, and F. Naumann. Latent topics in graph-structured data. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2663–2666, Maui, HI, 2012.

[9] C. Böhm, J. Lorey, and F. Naumann. Creating voiD descriptions for web-scale data. *Journal of Web Semantics*, 9(3):339–345, 2011.

[10] C. Böhm, F. Naumann, Z. Abedjan, D. Fenz, T. Grütze, D. Hefenbrock, M. Pohl, and D. Sonnabend. Profiling linked open data with ProLOD. In *Proceedings of the International Workshop on New Trends in Information Integration (NTII)*, pages 175–178, Long Beach, CA, 2010.

[11] L. Bravo, W. Fan, and S. Ma. Extending dependencies with conditions. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 243–254, Vienna, Austria, 2007.

[12] M. J. Cafarella, A. Halevy, and J. Madhavan. Structured data on the web. *Communications of the ACM*, 54(2):72–79, 2011.

[13] S. Chaudhuri, U. Dayal, and V. Ganti. Data management technology for decision support systems. *Advances in Computers*, 62:293–326, 2004.

[14] F. Chiang and R. J. Miller. Discovering data quality rules. *Proceedings of the VLDB Endowment*, 1:1166–1177, 2008.

[15] P. Christen. *Data Matching*. Springer Verlag, Berlin – Heidelberg – New York, 2012.

[16] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.

[17] O. Curé. Conditional inclusion dependencies for data cleansing: Discovery and violation

detection issues. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, Lyon, France, 2009.

[18] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer Verlag, Berlin – Heidelberg – New York, 2007.

[19] S. M. Fakhrahmad, M. H. Sadreddini, and M. Z. Jahromi. AD-Miner: A new incremental method for discovery of minimal approximate dependencies using logical operations. *Intelligent Data Analysis*, 12(6):607–619, 2008.

[20] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Transactions on Database Systems (TODS)*, 33(2):1–48, 2008.

[21] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 23(4):683–698, 2011.

[22] L. Golab, F. Korn, and D. Srivastava. Efficient and effective analysis of data quality using pattern tableaux. *IEEE Data Engineering Bulletin*, 34(3):26–33, 2011.

[23] L. Golab and M. T. Özsu. *Data Stream Management*. Morgan Claypool Publishers, 2010.

[24] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2011.

[25] D. I. Holmes. Authorship attribution. *Computers and the Humanities*, 28:87–106, 1994.

[26] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *Computer Journal*, 42:100–111, 1999.

[27] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 647–658, Paris, France, 2004.

[28] S. Kandel, R. Parikh, A. Paepcke, J. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of Advanced Visual Interfaces (AVI)*, pages 547–554, Capri, Italy, 2012.

[29] D. A. Keim and D. Oelke. Literature fingerprinting: A new method for visual literary analysis. In *Proceedings of Visual Analytics Science and Technology (VAST)*, pages 115 –122, Sacramento, CA, 2007.

[30] S. Lopes, J.-M. Petit, and F. Toumani. Discovering interesting inclusion dependencies: application to logical database tuning. *Information Systems*, 27(1):1–19, 2002.

[31] A. Löser, F. Hueske, and V. Markl. Situational business intelligence. In *Proceedings Business Intelligence for the Real-Time Enterprise (BIRTE)*, pages 1–11, Auckland, New Zealand, 2008.

[32] M. V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):191–221, 1988.

[33] F. D. Marchi, S. Lopes, and J.-M. Petit. Efficient algorithms for mining inclusion dependencies. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 464–476, Prague, Czech Republic, 2002.

[34] F. D. Marchi, S. Lopes, and J.-M. Petit. Unary and n-ary inclusion dependency discovery in relational databases. *Journal of Intelligent Information Systems*, 32:53–73, 2009.

[35] V. M. Markowitz and J. A. Makowsky. Identifying extended entity-relationship object structures in relational schemas. *IEEE Transactions on Software Engineering*, 16(8):777–790, 1990.

[36] T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice-Hall, 2nd edition, 1999.

[37] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 294–305, Montreal, Canada, 1996.

[38] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.

[39] E. Rahm and H.-H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[40] V. Raman and J. M. Hellerstein. Potters Wheel: An interactive data cleaning system. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 381–390, Rome, Italy, 2001.

[41] A. Rostin, O. Albrecht, J. Bauckmann, F. Naumann, and U. Leser. A machine

learning approach to foreign key discovery. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB)*, Providence, RI, 2009.

[42] Y. Sismanis, P. Brown, P. J. Haas, and B. Reinwald. GORDIAN: Efficient and scalable discovery of composite keys. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 691–702, Seoul, Korea, 2006.

[43] K. P. Smith, M. Morse, P. Mork, M. H. Li, A. Rosenthal, M. D. Allen, and L. Seligman. The role of schema matching in large enterprises. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, 2009.

[44] W. Wu, B. Reinwald, Y. Sismanis, and R. Manjrekar. Discovering topical structures of databases. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1019–1030, Vancouver, Canada, 2008.

[45] H. Yao and H. J. Hamilton. Mining functional dependencies from data. *Data Mining and Knowledge Discovery*, 16(2):197–219, 2008.