Constance: An Intelligent Data Lake System

Conference Paper · June 2016

DOI: 10.1145/2882903.2899389

3 authors:

Rihan Hai
RWTH Aachen University
12 PUBLICATIONS **95** CITATIONS
SEE PROFILE

Sandra Geisler
Fraunhofer Institute for Applied Information Technology FIT
34 PUBLICATIONS **225** CITATIONS
SEE PROFILE

Christoph Quix
Fraunhofer Institute for Applied Information Technology FIT
121 PUBLICATIONS **1,405** CITATIONS
SEE PROFILE

Constance: An Intelligent Data Lake System

Rihan Hai
RWTH Aachen University
Aachen, Germany
hai@dbis.rwth-aachen.de

Sandra Geisler
RWTH Aachen University
Aachen, Germany
geisler@dbis.rwth-aachen.de

Christoph Quix
Fraunhofer FIT
St. Augustin, Germany
christoph.quix@fit.fraunhofer.de

ABSTRACT

As the challenge of our time, Big Data still has many research hassles, especially the variety of data. The high diversity of data sources often results in information silos, a collection of non-integrated data management systems with heterogeneous schemas, query languages, and APIs. *Data Lake* systems have been proposed as a solution to this problem, by providing a schema-less repository for raw data with a common access interface. However, just dumping all data into a data lake without any metadata management, would only lead to a 'data swamp'. To avoid this, we propose *Constance*[1], a Data Lake system with sophisticated metadata management over raw data extracted from heterogeneous data sources. Constance discovers, extracts, and summarizes the structural metadata from the data sources, and annotates data and metadata with semantic information to avoid ambiguities. With embedded query rewriting engines supporting structured data and semi-structured data, Constance provides users a unified interface for query processing and data exploration. During the demo, we will walk through each functional component of Constance. Constance will be applied to two real-life use cases in order to show attendees the importance and usefulness of our generic and extensible data lake system.

1. INTRODUCTION

Data lakes address the daunting challenge of Big Data: 'how to make an easy use of highly diverse data and provide knowledge?' Large amounts of data are available, but often the data is separated in information silos without or only with loose inter-connections. However, valuable insights are often only available upon the combination and integrated analysis of information in these silos. To meet this gap, *Data Lakes* (DLs) have been conceptualized as big data repositories which store raw data and provide functionality for on-demand integration with the help of metadata descriptions [1, 3, 9, 10]. DLs ingest raw data in its original format from heterogeneous data sources, fulfill their role as storage repositories, and allow users to query and explore them. As schema information, mappings, and other constraints are not defined explicitly or required initially for a DL, it is important to extract as much metadata as possible from the data sources during the ingestion phase. Metadata management is crucial for data reasoning, query processing, and data quality management. Without any metadata, the DL is hardly usable as the structure and semantics of the data are not known, which turns a Data Lake quickly into a 'data swamp'.

However, current proposals for DL systems lack details about the required metadata management features and methods for efficient integrated query processing, which hampers repeatable implementations. Furthermore, the DL systems put into practice now are often very application-specific solutions which have been crafted for the specific requirements of an organization. These specific solutions are based on generic software frameworks such as Hadoop and still require a lot of (wo)man power for adaptation and customization of the generic frameworks.

Therefore, in this paper we propose Constance as a first step towards a practical DL solution which can be used as the basis in several DL projects, because it provides a flexible, extensible framework for the data management problems within DL systems. In particular, Constance is a DL system which manages structural and semantic metadata, provides means to enrich the metadata with schema matching and schema summarization techniques, and offers a unified interface for query processing. To emphasize the importance of metadata in the maturation of a DL, in this demo we primarily focus on structured and semi-structured data accompanied by either explicit or implicit metadata. In contrast to other DL proposals [10, 2], Constance focuses more on data ingestion, metadata management, and query answering, but also provides basic security and provenance mechanisms. Before we go into more details of our proposal, we want to clarify two key points regarding the current DL debate [4]. First, a DL is more than just a storage repository on top of a Hadoop file system. Of course, Hadoop is good at managing the huge amount of data in a DL with its distributed and scalable file system; it also provides metadata management functions, but it does not provide all the metadata functionalities which are required for a DL. For

[1]Our system is named after Lake Constance, the largest lake in Germany (known as Bodensee in German). Shared by Switzerland, Germany and Austria, Lake Constance is an area with no borders, which is similar to our design philosophy for an open Data Lake.

SIGMOD '16, June 26–July 1, 2016, San Francisco, CA, USA.

 ISBN 978-1-4503-3531-7/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2882903.2899389

example, the DL architecture presented in [2] shows that a DL system is a complex eco-system of several components and that Hadoop provides only a part of the required functionality. Second, compared to data warehouse systems with a relational view of data, DLs need to handle more heterogeneous data sources including semi-structured and unstructured sources. Since the NoSQL movement introduced several new data models in the recent years, a DL system has to provide a flexible storage system which can handle these different data models efficiently in an integrated way.

Putting these ideas together, we propose in this paper *Constance*, which features the following concepts: (i) Constance provides a **metadata management system** which extracts explicit and implicit metadata from the imported data. (ii) Composed of semantic annotations, advanced modeling, and record linkage, Constance contributes **semantic metadata matching and enrichment**, therefore it improves the quality of data stored in the DL. (iii) For an easy use by both data experts and naive users, Constance supports a **formal structured query language** as well as simple **keyword queries**. Keyword queries are translated into a formal query by using a query formulation function. (iv) Constance provides an **integrated user interface**, which allows users to check the visualized impact of every schema management step, to query the data, and to monitor the **data quality** of the DL by user-defined quality metrics.

The rest of the paper is organized as follows: In Sec. 2, we give a system overview of Constance and its essential components. We present a brief walkthrough of a Constance workflow, combined with two real-life use cases in Sec. 3.

2. CONSTANCE OVERVIEW

Fig. 1 depicts the architecture of Constance, as well as its key components. Constance can be roughly divided into three functional layers: ingestion, maintenance, and querying. While the ingestion layer implements the interface between the data sources and Constance, the major human-machine interaction is enabled by the querying layer. In the middle are the components constituting the maintenance layer, which dynamically and incrementally extract and summarize the current metadata of the DL, and provide a uniform query interface.

2.1 Ingestion

The ingestion layer is responsible for importing data from heterogeneous sources into the DL system. Different from the traditional ETL process, Constance loads data in its original format without forcing expensive transformation or integration tasks. As shown in Fig. 1, regardless of the format of the source data (e.g., JSON, spreadsheets, XML or relational data), data is loaded and stored in Constance in its original format. Notably, for further metadata processing in the maintenance layer of the system, Constance extracts metadata from the data sources into a unified metamodel.

2.2 Maintenance

The components in this layer mainly contribute to Constance's metadata management functions. The backend of the maintenance layer provides the basic functions for data storage and efficient querying. Metadata is crucial for future querying. The first step of metadata management is extracting as much metadata as possible from the sources. For relational tables and some XML documents, explicit

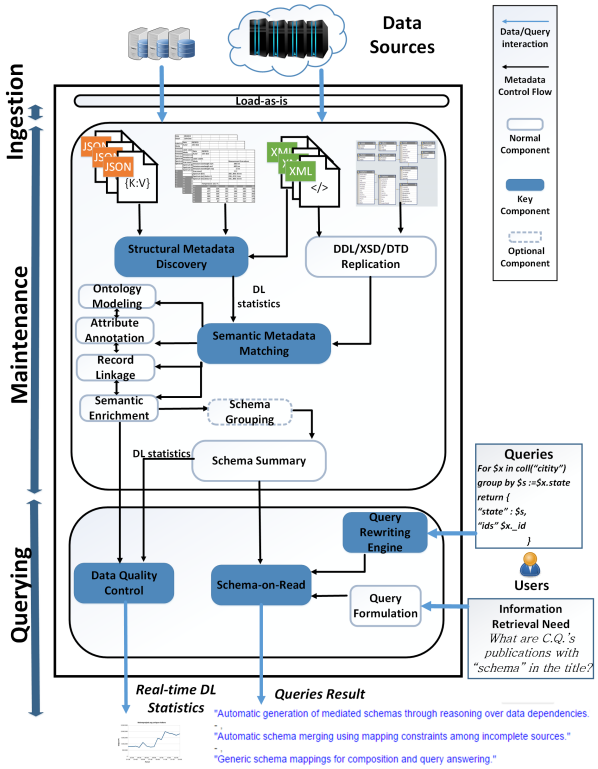

Figure 1: Constance System Overview

schema definitions (in SQL, XSD, or DTD) can be directly obtained from the source and integrated into the metamodel. The tricky part is semi-structured data (such as XML without XSD, JSON, or partially structured Excel or CSV files) which contains implicit schemata. Therefore, the component *Structural Metadata Discovery (SMD)* takes over the responsibility of discovering implicit metadata (e.g., entity types, relationship types, and constraints) from semi-structured data in a two-fold manner: (1) The SMD checks whether metadata is encoded inside the raw data file, in the filename, or the directories of the file system. An example is the self-describing spreadsheet in Fig. 1, where SMD extracts its schema (headers in columns and rows). (2) For semi-structured documents such as JSON or XML, SMD discovers "Has-a"-relationships and stores them in a path-like structure. Notably, as JSON records may not have sufficient relationships among documents, during run time Constance calculates frequencies of the entities which appeared in join conditions of user queries, and marks them as potential foreign key relationships and builds indexes on the corresponding attributes to improve the performance. This is the part we refer as 'incrementally maturing of a DL' as the metadata will be enriched in a pay-as-you-go fashion while users are using the system.

Another key component of Constance is the *Semantic Metadata Matching (SMM)* component, which consists of ontology modeling, attribute annotation, record linkage, and semantic enrichment. The output of this component can be observed in Fig. 2.b, which is a graph representation of the extracted metadata elements and their relationships. Semantic annotations about the meaning of certain metadata items can be added to the schema elements, and therefore, they can be exploited for querying. Using the semantic an-

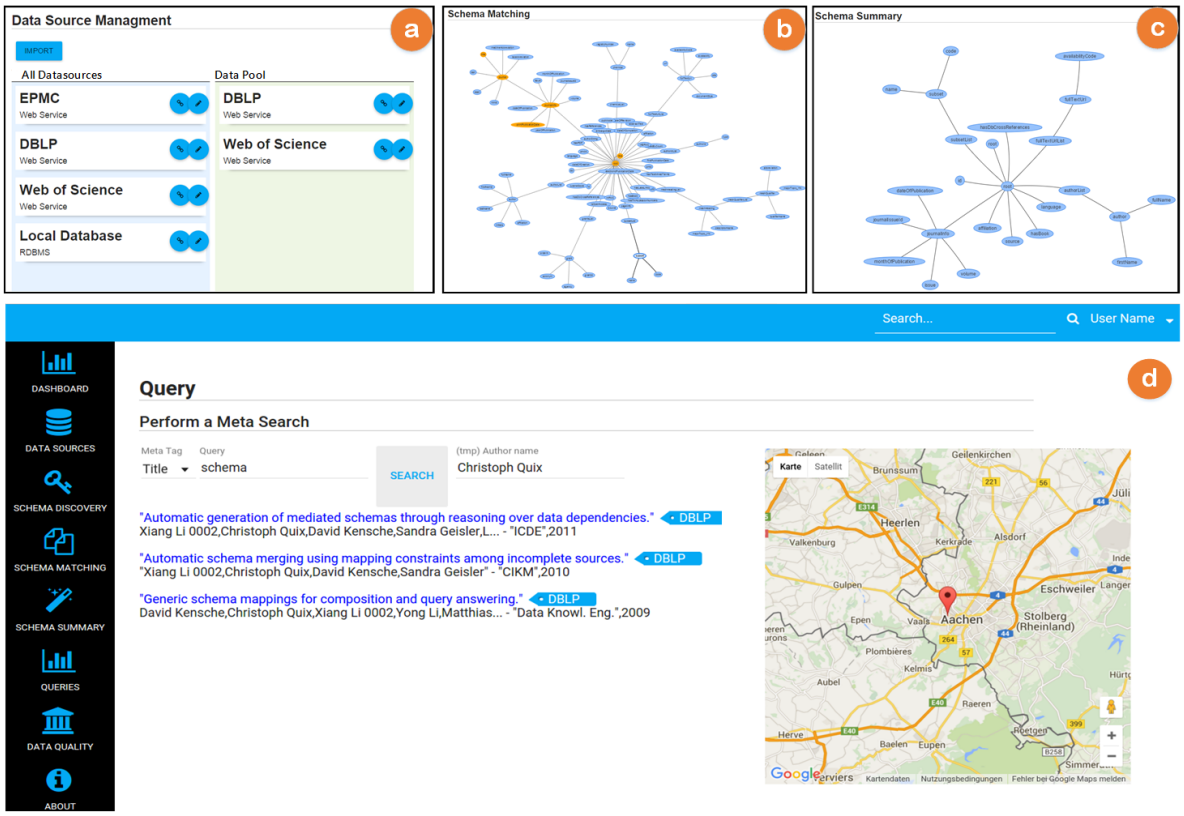

Figure 2: Major components of Constance user interface: (a) Data source management (b) Schema matching (c) Schema summary (d) Query Answering

notations, we perform a first shallow integration of the data as metadata elements with different labels (e.g., attributes 'dob' and 'DateOfBirth') can be linked if they are annotated with the same semantic element (e.g., 'foaf:birthday'). The user can then use the semantic term 'foaf:birthday' in a query; the query rewriting module of Constance rewrites the user query into a union of queries which use the schema elements from the sources. Semantic annotations are done with standardized domain-specific models or ontologies which provide a common, shared understanding for the domain.

As an optional step, *Schema Grouping* clusters the schemas and picks up the core of each cluster as its presentation. The necessity of grouping depends on the schema similarity calculated over the imported data sources. If the schemas are very distinctive or almost the same, schema grouping would not contribute much to the overall system improvement.

Based on the enriched and potentially grouped metadata, the *Schema Summary* component extracts a further compact structure of the currently managed schemata, which we refer to as *skeleton* [11], sketched in Fig. 2.c. Summarizing schemata is a method to filter the most important elements or relationships, respectively, within a complex schema. Elements and relationships used more frequently in instance data or queries are considered as more important.

2.3 Query Answering

All the above functions, eventually, serve for information retrieval, in the form of query answering. In common cases, users either input concrete queries in a particular query language, or they have an information retrieval need which supports the user in formulating a query starting from some keywords. Constance distributes the user need to the *Query Rewriting Engine* or *Query Formulation*, respectively. Constance currently implements a query language which is a subset of JSONiq (a query language for JSON-based systems which is similar to XQuery [5]) as this enables easy querying of semi-structured data in JSON. The semantic annotations mentioned above can be seen as simple global-as-view mappings, which means that a single user query might be rewritten into a union of queries over the sources in which the semantic terms are replaced with the original schema elements of the sources. We currently work on a more complex mapping and query language which is based on our previous work on second-order tuple generating dependencies for mappings between generic data models [8].

If the user prefers to start with keywords, the *Query Formulation* component supports users in creating formal queries that express the intended information requirement. The user will see only a simple natural language representation of the query. The entered values will be matched with metadata and data values. Depending on the matched items, the system will propose further related (meta)data items (e.g., if 'publication' has been recognized, the system might propose 'author' and 'title'). The goal is to have in the end a formal query expressed in our JSONiq-subset.

2.4 Data Quality Control

In addition to the advanced metadata management and querying functions, Constance provides also means for data quality management. The data quality model is based on our previous works for data warehouses [7] and data streams [6]. We define quality factors based on metrics which perform data quality measurements by executing queries (which return some aggregate value) or by invoking some specific functions (implemented as Java methods). The definition of quality metrics is application-specific and can be defined by the user for his/her DL instance. The values are presented in a summarized form in a data quality dashboard.

2.5 Extensibility and Flexibility

So far we have introduced the generic components of Constance. To apply it in real life projects, scenario-specific functionalities have to be integrated. Mostly notably in the raw data ingestion, the extractor components have to be adapted to some concrete data sources. We have developed an extensible and flexible framework for the ingestion phase in which the existing generic extractors can be adapted to new data source formats (e.g., by providing patterns for the data to be extracted from CSV or Excel files) or new extractors can be easily added using a plug-in mechanism.

On the querying side, the users might expect special ways of data visualization or data interaction of the query results. For example, as indicated in Fig. 2.d, fields referring to a geographic location (in this case, the author's affiliation) may be visualized in a map.

3. DEMONSTRATION WALKTHROUGH

Constance is a web-based application which follows Google's material design[2] for better user experience. Users can click on the button menu in the left, and switch among the key components, as shown in Fig. 2. To showcase the usefulness of Constance, we will demonstrate a typical workflow of Constance, derived from the medical engineering project *mi-Mappa*[3] which requires the integration of patent and publication data. Another use case is the integration of semi-structured data in the life science domain, which is currently being developed in the HUMIT project[4].

The demonstration will include the following scenarios:

Data Source Import Our demonstration begins with the data import. As shown in Fig. 2.a, users can import local files, databases, or remote data via web services. The imported data stores are shown in the left area as *All Data Sources* in the DL. Considering a running DL may contain a huge amount of data sources, while not all contribute to a certain analytical operation, users can select a subset of the data sources for the following step. This can be achieved by dragging and dropping the data sources from the left area to the right area 'data pool'. The selection can be also supported by queries over the data source metadata.

Schema Management Workflow By entering the Schema Discovery page, the schema discovery process is automatically performed and the extracted schema is visualized for the user. In the following Schema Matching step, Constance detects and equivalently maps entity types while users can see highlighted nodes, as shown in Fig. 2.b. Semantically equivalent entities, which come from different data sources and may vary by naming or structure, are replaced with one node and marked as yellow. After the execution of Schema Summary, a smaller set of nodes, representing a more concise set of the overall schema, is depicted in Fig. 2.c.

Querying In this section we will demonstrate how Constance retrieves information from its internal storage repository which has been filled by the data sources. As stated above, Constance provides two main options for querying, either using formal queries or keyword queries. The demo will show how these two different types of queries are processed in the system, how the queries are rewritten, and which queries will be finally evaluated over the internal data repository. Furthermore, we will show the possibilities to explore and browse the data as well as the metadata repository.

[2]https://www.google.com/design/spec/material-design/
[3]http://dbis.rwth-aachen.de/cms/projects/mi-mappa
[4]http://www.humit.de

4. ACKNOWLEDGMENTS

This work has been funded partially by the Klaus-Tschira-Stiftung (project mi-Mappa, http://www.dbis.rwth-aachen.de/mi-Mappa/, project no. 00.263.2015) and by the German Federal Ministry of Education and Research (BMBF) (project HUMIT, http://humit.de/, grant no. 01IS14007A) We would like to thank Patrick Schlesiona and David Kensche for their support and comments.

5. REFERENCES

[1] H. Alrehamy and C. Walker. Personal data lake with data gravity pull. In *Proc. IEEE BDCloud*, 2015.
[2] E. Boci and S. Thistlethwaite. A novel big data architecture in support of ads-b data analytic. In *Proc. ICNS*, 2015.
[3] M. Chessell, F. Scheepers, N. Nguyen, R. van Kessel, and R. van der Starre. Governing and managing big data for analytics and decision makers. www.redbooks.ibm.com/redpapers/pdfs/redp5120.pdf, 2014.
[4] J. Dixon. Data lakes revisited. https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/, September 2014.
[5] D. Florescu and G. Fourny. Jsoniq: The history of a query language. *IEEE Internet Computing*, 17(5):86–90, 2013.
[6] S. Geisler, S. Weber, and C. Quix. An ontology-based data quality framework for data stream applications. In *Proc. ICIQ*, 2011.
[7] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Information Systems*, 24(3):229–253, 1999.
[8] D. Kensche, C. Quix, X. Li, Y. Li, and M. Jarke. Generic schema mappings for composition and query answering. *Data Knowl. Eng.*, 2009.
[9] C. Quix, R. Hai, and I. Vatov. Gemms: A generic and extensible metadata management system for data lakes. In *CAISE FORUM*, 2016.
[10] I. Terrizzano, P. M. Schwarz, M. Roth, and J. E. Colino. Data wrangling: The challenging yourney from the wild to the lake. In *Proc. CIDR*, 2015.
[11] L. Wang, S. Zhang, J. Shi, L. Jiao, O. Hassanzadeh, J. Zou, and C. Wangz. Schema management for document stores. *PVLDB*, 8(9):922–933, 2015.