Injecting Conceptual Constraints into Data Fabrics

Paolo Ciaccia[1], Davide Martinenghi[2] and Riccardo Torlone[3]

[1]*Dipartimento di Informatica - Scienza e Ingegneria, Università di Bologna, Italy*
[2]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy*
[3]*Dipartimento di Ingegneria, Università Roma Tre, Italy*

Abstract
Unlike traditional sources managed by DBMSs, data lakes do not provide any guarantee about the quality of the data they store, which can severely limit their use for analysis purposes. The recent notion of data fabric, which introduces a semantic layer allowing uniform access to underlying data sources, makes it possible to tackle this problem by specifying conceptual constraints to which data sources must adhere to be considered meaningful. Along these lines, in this discussion paper, we exploit the data fabric approach by proposing a general methodology for data curation in data fabrics based on: (i) the specification of integrity constraints over a conceptual representation of the data lake and (ii) the automatic translation and enforcement of such constraints over the actual data. We discuss the advantages of this idea and the challenges behind its implementation.

1. Introduction

In traditional big data analysis, activities such as cleaning, transforming, and integrating source data are essential but they usually make knowledge extraction a very long and tedious process. For this reason, data-driven organizations have recently adopted an agile strategy that dismisses any data processing before their actual consumption. This is done by building and maintaining a repository, called “data lake”, for storing any kind of data in its native format. A dataset in the lake is usually just a collection of raw data, either gathered from internal applications (e.g., logs or user-generated data) or from external sources (e.g., open data), that is made persistent on a storage system, usually distributed, “as is”, without going through an ETL process.

Unfortunately, reducing the engineering effort upfront just delays the traditional issues of data pre-processing since this approach does not eliminate the need for high-quality data and schema understanding. Therefore, to guarantee reliable results, a long process of *data preparation* (a.k.a. *data wrangling*) is required over the portion of the data lake that is relevant for a business purpose before any meaningful analysis can be performed on it [1, 2, 3, 4]. This process typically consists of pipelines of operations such as: source and feature selection, data enrichment, data transformation, data curation, and data integration. A number of state-of-the-art applications can support these activities, including (i) data and metadata catalogs, for understanding and selecting the appropriate datasets [5, 6, 7, 8]; (ii) tools for full-text indexing, for providing keyword search and other advanced search capabilities [7, 9, 10]; (iii) data profilers,

SEBD 2023: 31st Symposium on Advanced Database Systems, July 02–05, 2023, Galzignano Terme, Padua, Italy
paolo.ciaccia@unibo.it (P. Ciaccia); davide.martinenghi@polimi.it (D. Martinenghi); riccardo.torlone@uniroma3.it (R. Torlone)

for collecting meta-information from datasets [1, 9, 11]; (iv) distributed data processing engines like Spark [12], and (v) tools and libraries for data manipulation and analysis, such as Pandas[1] and Scikit-learn,[2] in conjunction with data science notebooks, such as Jupyter[3] and Zeppelin.[4] Still, data preparation is an involved, fragmented, and time-consuming process, thus making the extraction of valuable knowledge from the lake hard.

In this scenario, the recent data fabric approach comes to the rescue, by proposing the construction and maintenance of a semantic representation of the underlying data for data discovery, understanding, and searching [13, 14, 15]. We argue that this can also be profitably exploited for evaluating and improving the quality of data. This is because a representation of the real-world concepts and relationships that the data capture (e.g., employees, customers, products, locations, sales, and so on) provides an ideal setting for identifying the constraints that hold in the application domain of reference (e.g., the fact that, for business purposes, all the products for sale must be classified in categories). If we are able to map and enforce such constraints on the underlying data, their quality naturally improves and makes the subsequent analysis more effective and less prone to errors.

Building on this idea, in this paper[5] we propose a principled approach to data curation in data lakes based on the identification and enforcement of conceptual constraints. The approach is based on the following main activities: (1) the gathering of metadata from the data lake (or from a portion of interest for a specific business goal) in the form of a conceptual schema, (2) the analysis of the conceptual schema and the specification of integrity constraints over it, (3) the automatic translation of the constraints defined at the conceptual level into constraints over the datasets in the data lake, (4) the enforcement of the integrity constraints so obtained over the actual data. While there is a large body of works on extracting and collecting metadata from data sources [1, 9, 11] and on repairing data given a set of integrity constraints [17, 18, 19], corresponding to steps (1) and (4) above, to our knowledge the issue of exploiting conceptual representations for data lake curation has never been explored before.

The rest of the paper is devoted to the presentation of some initial steps towards this goal.

Specifically, in Section 2 we state the problem by recalling the typical data life-cycle in a data lake and by illustrating, in this framework, our proposal for data curation. Then, in Section 3 we state the basic notions (datasets, schemas, constraints, and mappings) underlying our approach. This is done by means of very general definitions, in order to make the approach independent of any specific data model and format. In Section 4 we provide some details of our solution through an example. Finally, in Section 5 we discuss the related works, the main issues involved in the implementation of our proposal, and the work that needs to be done to tackle these issues.

2. Using Conceptual Constraints in Data Curation

Metadata plays a fundamental role in typical activities part of the life-cycle of data analysis, such as data ingestion, data integration, data preparation, and knowledge extraction. Its management

[1]https://pandas.pydata.org/
[2]https://scikit-learn.org/
[3]https://jupyter.org/
[4]https://zeppelin.apache.org/
[5]A preliminary version has appeared in [16].

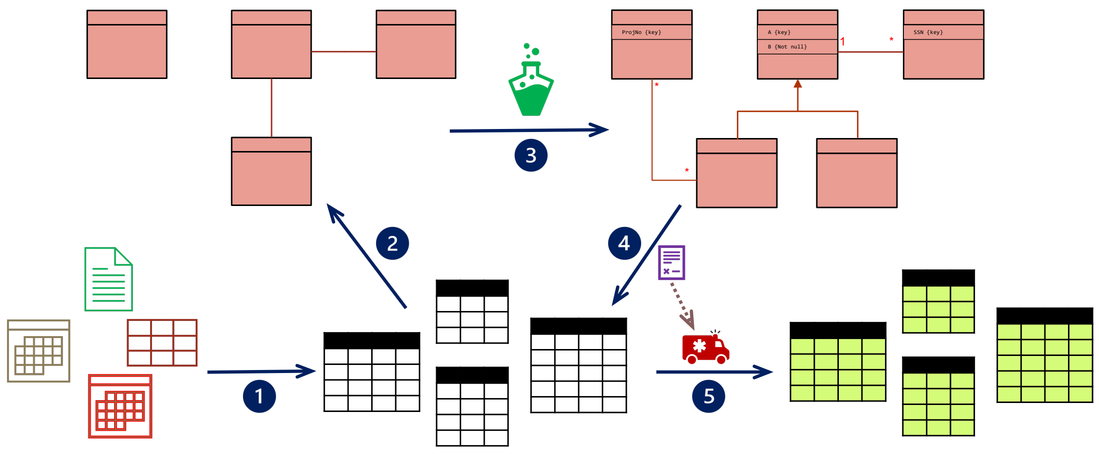

Figure 1: The process of data curation through conceptual constraints.

involves building and maintaining a repository of information describing all the various kinds of data that are produced in the above stages of data processing [9].

In order to harmonize the data stored in the data lake with the other components of the data fabric, a conceptual representation, i.e., a *conceptual schema*, of the metadata describing the content of interest of the data lake is needed. This includes concepts (such as entities, relationships, and generalizations) that map to the actual components (such as attributes, documents, and labels) of datasets stored in the data lake.

The availability of a conceptual schema $\mathcal{S}$ of data lake $\mathcal{D}$ can provide a number of important benefits: *(i)* it allows the analysts to have a general and system-independent vision of the data available in $\mathcal{S}$, *(ii)* it provides an abstract view of the data lake content which can be used to define and possibly specify queries over $\mathcal{D}$, and *(iii)* it allows the specification of real-world constraints that, enforced on $\mathcal{D}$, improve the overall quality of its content.

In this paper, we focus on Problem *(iii)* above that, to the best of our knowledge, has not been studied before, apart from our preliminary study [16]. As shown graphically in Figure 1, the methodology we propose requires the tasks that follow.

1. A (portion of interest of a) data lake $\mathcal{D}$ is initially transformed into a "standardized" format, obtained by adapting source data to that of the system chosen for storing a "curated" version of $\mathcal{D}$.
2. The *skeleton* $\widehat{\mathcal{S}}$ of a conceptual schema is built from $\mathcal{D}$. Basically, $\widehat{\mathcal{S}}$ includes the main entities and relationship involved in $\mathcal{D}$ as well as a mapping between the components of $\mathcal{D}$ and the elements of $\widehat{\mathcal{S}}$. This task can be done manually and/or using available techniques and tools for semantic annotation or column-type discovery in data lakes [20, 21, 22].
3. $\widehat{\mathcal{S}}$ is refined, possibly incrementally, into an "evolved" schema $\mathcal{S}$ by adding a collection of real-world constraints. For instance, by stating that an entity is a special case of another entity or that an entity can only participate in a single occurrence of a certain relationship. Typically, this step requires knowledge of the specific domain (e.g., that a department has a single manager).

Finance Department
```
{"DeptCode":"D01", "DeptName":"Finance", "MgrNo":"E05", "employees": [
 {"EmpNo":"E05", "Name":"Homer", "Salary":"100K", "Level":3.5},
 {"EmpNo":"E12", "Name":"Lisa", "Salary":"50K", "CV":"My...",
 "activities": [
 {"Activity":"Research", "NoHours":50}
 ]
]
```
Text Department
```
{"DeptCode":"D02", "DeptName":"Tech", "MgrNo":"E10", "employees": [
{ "EmpNo":"E07", "Name":"Marge", "Salary":"150K", "Level":4,
 { "CV":"Hi!..."},
 { "EmpNo":"E10", "Name":"Bart", "Salary":"80K", "CV":"I'm...",
 { "PID":"P01", "PName":"iScream", "Budget":"10M",
 "activities": [
 {"Activity":"Reporting", "NoHours":30},
 {"Activity":"Research", "NoHours":150}
 ]}
]}
```
Figure 2: Raw data available as JSON-formatted files.

4. The constraints represented by $\mathcal{S}$ are mapped to constraints $\mathcal{C}_{\mathcal{D}}$ over the actual data stored in $\mathcal{D}$. $\mathcal{C}_{\mathcal{D}}$ can be expressed in several ways, depending on the system used to store and manage $\mathcal{D}$.
5. The constraints $\mathcal{C}_{\mathcal{D}}$ are checked on $\mathcal{D}$ and, possibly, enforced on $\mathcal{D}$ by means of a *repairing* technique [23], if any violation occurs. Again, this can be done in several ways, depending on the tools available for storing, querying, and manipulating data in the data lake [19, 24].

We can notice that in the process above no specific work has specifically addressed point 4. In the rest of the paper, we focus on this challenging task by first introducing the relevant elements of the problem (Section 3), and by then illustrating the main ideas for its solution through an example (Section 4).

3. Data and Metadata Management

Let us now fix some basic notions that we will refer to in the following. Our definitions are deliberately abstract so as to be as general as possible, without the need to commit to any specific data lake model and format.

Dataset. We consider that a dataset $DS(X, D)$ has a name DS and is composed of a set X of *attributes* and a set D of *data items*. Each data item in D is a set of attribute-value pairs, with attributes taken from X. Figure 2 shows an example of datasets still in a "raw" format, reporting data about the finance and tech departments of a company. After curation, the so-obtained datasets also take part in the data lake.

Data Lake. For our purposes, a data lake $DL = (\mathcal{D}, \mathcal{M})$ can be modeled as a collection $\mathcal{D}$

D_Emp

EmpNo	Name	Salary	DeptCode	Level	CV	PID	PName	Budget
E05	Homer	100K	D01	3.5	-	-	-	-
E07	Marge	150K	D02	4	“Hi!...”	-	-	-
E10	Bart	80K	D02	-	“I’m...”	P01	iScream	10M
E12	Lisa	50K	D01	-	“My...”	-	-	-

D_Dept

DeptCode	DeptName	MgrNo
D01	Finance	E05
D02	Tech	E10

D_Act

ResNo	Activity	NoHours
E10	Reporting	30
E10	Research	150
E12	Research	50

Figure 3: Datasets in the curated layer of a data lake.

of datasets having distinct names, plus a set of metadata $\mathcal{M}$, including a (possibly empty) set of constraints $\mathcal{C}$ on the datasets. We also refer to $\mathcal{D}$ as the instance of DL. Figure 3 shows a collection of partially curated datasets in $\mathcal{D}$ (D_Emp, D_Dept, and D_Act) that have been obtained from the raw datasets of Figure 2 by unnesting employees from departments and activities from employees. The metadata include, e.g., cross-dataset constraints, such as the fact that DeptCodes appearing in D_Emp must also appear in D_Dept, as well as, say, domain constraints such as the fact that Level must be an integer (so employee E_05 violates this).

Conceptual schema and constraints. We consider that the domain of interest for analysis purposes is represented by a *conceptual schema* $\mathcal{S}$, expressed by means of a suitable language $\mathcal{L}_{\mathcal{S}}$. Examples are Entity-Relationship (E-R) diagrams, RDF(S), UML’s class diagrams, and Description Logic languages, such as those underlying the OWL 2 standard and its profiles.[6]. Besides specific differences, each of these languages allows for the definition of *concepts* (i.e., classes of objects, entities), *relationships* (a.k.a. as *roles)* among them, and *properties* (of concepts and relationships). A conceptual schema includes *conceptual constraints* that characterize the elements of the schema $\mathcal{S}$. For instance, in the E-R formalism we can state that two entities E_1 and E_2 have a common generalizing entity E (subset(E_1, E) and subset(E_2, E)) and that E_1 and E_2 are disjoint (disjoint(E_1, E_2)).

Mapping. The connection between the conceptual schema $\mathcal{S}$ and the data lake $(\mathcal{D}, \mathcal{M})$ is based on a mapping μ, i.e., a set of assertions relating the elements in $\mathcal{S}$ to the datasets in $\mathcal{D}$. For instance, an entity Departments in $\mathcal{S}$ could be mapped to the projection of dataset D_Dept on just the attributes DeptCode and DeptName, with the MgrNo attribute representing a relationship between Departments and Employees.

Problem statement. Our goal is to check whether an instance $\mathcal{D}$ satisfies the conceptual constraints represented by schema $\mathcal{S}$. To this end, we formally define constraint satisfaction as follows.

Definition 1. *An instance $\mathcal{D}$ is <u>legal</u> with respect to a conceptual schema $\mathcal{S}$ through a mapping*

[6]https://www.w3.org/TR/owl2-profiles/

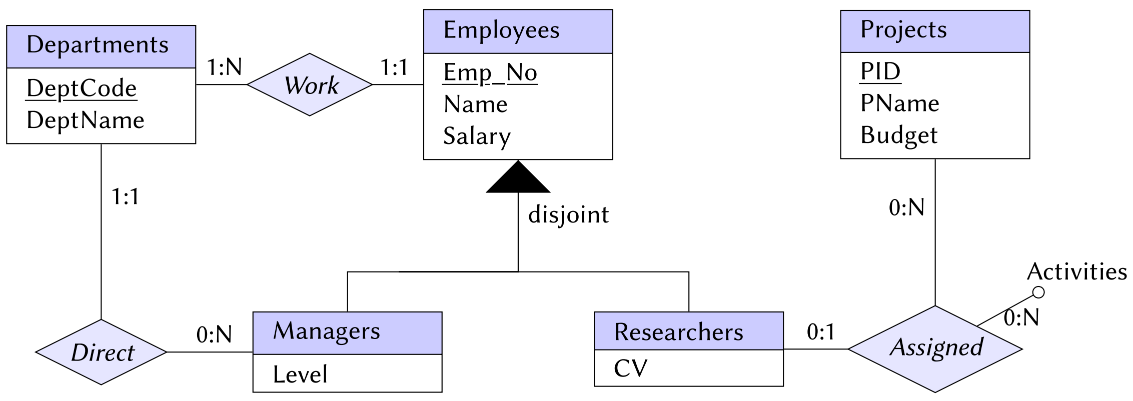

Figure 4: A conceptual schema $\mathcal{S}$.

μ if $\mu(\mathcal{D})$ yields a conceptual instance that satisfies all the constraints in $\mathcal{S}$, and shortly indicate this circumstance as $\mathcal{S} \models \mu(\mathcal{D})$.

Clearly, explicitly applying the mapping to generate a conceptual instance is impractical for performance reasons. Therefore, in this paper we propose a different solution, which essentially consists in transforming the constraints in $\mathcal{S}$ into corresponding constraints on the data lake. This leads to the following problem:
given a data lake $DL = (\mathcal{D}, \mathcal{M})$, a conceptual schema $\mathcal{S}$, a mapping μ between $\mathcal{S}$ and DL, **determine** a set of constraints $\mathcal{C}_{\mathcal{D}}$ on DL such that $\mathcal{D}$ satisfies $\mathcal{M} \cup \mathcal{C}_{\mathcal{D}}$ if and only if $\mathcal{D}$ is legal with respect to $\mathcal{S}$ through mapping μ, i.e.: $\mathcal{M} \cup \mathcal{C}_{\mathcal{D}} \models \mathcal{D} \iff \mathcal{S} \models \mu(\mathcal{D})$.

Once the conceptual constraints on the data lake $\mathcal{D}$ have been generated, they may be used to check if $\mathcal{D}$ is consistent and, eventually, to repair $\mathcal{D}$.

Before proceeding, we remark that, unlike OBDA (Ontology-Based Data Access) approaches [25], we do not use μ for the purpose of obtaining results from $\mathcal{D}$ given a query on $\mathcal{S}$. Rather, μ is the key ingredient to define and enforce on the data lake the conceptual constraints in $\mathcal{S}$.

4. An Example

The E-R schema $\mathcal{S}$ in Figure 4 describes a simplified scenario regarding the departments of a company. The schema includes structural information (such as the fact that Employees have a Name and a Salary) as well as constraints (such as the fact that Managers are also Employees or that each Department has at least one Employee). Notice that the schema $\mathcal{S}$ deliberately does not include the `NoHours` attribute that characterizes each activity of a researcher (see dataset `D_Act` in Figure 3). This is to emphasize that $\mathcal{S}$ only focuses on that part of the data lake that is of interest for the analysis, which here does not include, as we assume, the `NoHours` attribute.

Besides basic constraints on attributes, such as non-nullability and domain of admitted values (which, in the following, we will omit for brevity), relevant constraints in $\mathcal{S}$, here informally described as self-explanatory predicates, are:

`unique(EmpNo,Employees)`	every employee is identified by `EmpNo`
`unique(DeptCode,Departments)`	every department is identified by `DeptCode`
`. . .`	
`subset(Managers,Employees)`	managers are employees
`subset(Researchers,Employees)`	researchers are employees
`disjoint(Managers,Researchers)`	no manager is a researcher
`card(Departments,Direct,1,1)`	every department has exactly one manager
`card(Employees,Work,1,1)`	every employee works in exactly one department
`card(Departments,Work,1,n)`	every department has at least one employee

Now, consider the datasets in Figure 3, whose structure is reported below for the sake of clarity:

```
D_Emp(EmpNo,Name,Salary,DeptCode,Level,CV,PID,PName,Budget),
D_Dept(DeptCode,DeptName,MgrNo),
D_Act(ResNo,Activity,NoHours).
```

Then, we can define the mapping μ by means of the following statements, one for each entity and relationship in $\mathcal{S}$:[7]

```
 Employees(EmpNo,Name,Salary) :- D_Emp(EmpNo,Name,Salary,_,_,_,_,_,_)
        Managers(EmpNo,Level) :- D_Emp(EmpNo,_,_,_,Level,_,_,_,_),
                                 NotNull(Level)
        Researchers(EmpNo,CV) :- D_Emp(EmpNo,_,_,_,_,CV,_,_,_),
                                 NotNull(CV)
Departments(DeptCode,DeptName) :- D_Dept(DeptCode,DeptName,_)
    Projects(PID,PName,Budget) :- D_Emp(_,_,_,_,_,_,PID,PName,Budget)
       Direct(DeptCode,MgrNo) :- D_Dept(DeptCode,_,MgrNo)
         Work(DeptCode,EmpNo) :- D_Emp(EmpNo,_,_,DeptCode,_,_,_,_,_)
     Assigned(EmpNo,PID,Acts) :- D_Emp(EmpNo,_,_,_,_,_,PID,_,_),
                                 NotNull(PID),
                                 Acts = {Act | D_Act(EmpNo,Act,_)}
```

The constraints $\mathcal{C}_\mathcal{D}$ corresponding to this mapping, include, among others, the following ones:

- Uniqueness of `DeptCode`:
  $c_1 : \forall t_1, t_2 \in \texttt{D\_Dept} : t_1.\texttt{DeptCode} = t_2.\texttt{DeptCode} \Rightarrow t_1 = t_2$
- Disjointness of managers and researchers:
  $c_2 : \forall t_1 \in \texttt{D\_Emp} : \neg(\mathsf{NotNull}(t_1.\texttt{Level}) \wedge \mathsf{NotNull}(t_1.\texttt{CV}))$
- Departments are directed by managers:
  $c_3 : \forall t_1 \in \texttt{D\_Dept} \exists t_2 \in \texttt{D\_Emp} : t_1.\texttt{MgrNo} = t_2.\texttt{EmpNo} \wedge \mathsf{NotNull}(t_2.\texttt{Level})$
- Each department has at least one employee:
  $c_4 : \forall t_1 \in \texttt{D\_Dept} \exists t_2 \in \texttt{D\_Emp} : t_1.\texttt{DeptCode} = t_2.\texttt{DeptCode}$
- Each employee has activities only within a project:
  $c_5 : \forall t_1 \in \texttt{D\_Act} \exists t_2 \in \texttt{D\_Emp} : t_1.\texttt{ResNo} = t_2.\texttt{EmpNo} \wedge \mathsf{NotNull}(t_2.\texttt{PID})$

[7]The underscore symbol indicates (anonymous) variables not relevant to the statement. The adopted notation is therefore positional like in, e.g., Datalog.

Once the constraints in $\mathcal{C}_\mathcal{D}$ have been generated, they can be easily converted into proper queries so as to detect possible violations. For instance, constraint c_3 corresponds to the following query expressed in SQL:

```
SELECT D.DeptCode, E.EmpNo  FROM D_Dept D, D_Emp E
WHERE  D.MgrNo = E.EmpNo AND E.Level IS NULL
```

Consider now the datasets in Figure 3. It is apparent that $\mathcal{D}$ violates the following conceptual constraints in $\mathcal{C}_\mathcal{D}$:

- Employee `E07` has both attributes `Level` and `CV` not null, thus violating constraint c_2;
- Department `D02` is managed by an employee (`E10`) that is not a manager, contradicting constraint c_3, as the above SQL query would reveal;
- Constraint c_5 is also violated, since employee `E12` appears in the dataset `D_Act` although she does not participate in any project.

Once the above violations are discovered, the datasets can be cleaned using some of the available methods (see, e.g., [19] and [24]).

5. Discussion and Conclusions

In this paper we have put forward the idea of generating constraints on the datasets of a data lake by exploiting a high-level, conceptual representation, in order to improve the quality of data and, consequently, that of subsequent analysis.

Our approach can be regarded as complementary to those that aim to curate data by directly specifying constraints through ad-hoc languages/tools. For instance, CLAMS [24] adopts the RDF data model for representing data in the curated layer, and defines *conditional denial constraints* over views of the data lake defined using SPARQL queries. Although this is a powerful approach, able to exploit the expressivity of SPARQL, it leaves the full burden of specifying constraints (and queries) to the designer/analyst. Furthermore, there is no guarantee that the set of constraints is *consistent*, i.e., non-contradictory. The Deequ system [26, 27] is an open-source library aimed at supporting the automatic verification of data quality. However, the constraints available in the library apply to a single dataset, thus *inter-dataset constraints* cannot be specified.

A major challenge of our approach is to demonstrate that the propagation of conceptual constraints, i.e., the generation of $\mathcal{C}_\mathcal{D}$, can be fully automated. Although in the past decades a large body of work has investigated how to automatically translate ER schemas to relational tables (see, e.g., [28]), much less is known for other conceptual models and/or data models such as RDF. Our view of the problem currently considers (automatic) constraint propagation as a *two-step* process: (1) first, one operates a *canonical* transformation of the conceptual schema $\mathcal{S}$ into a schema $\mathcal{D}_{can}$ in the target data model of the curated layer; (2) then, $\mathcal{D}_{can}$ is mapped to the actual $\mathcal{D}$. Besides the obvious advantage of splitting the complexity of the problem into two well-defined sub-problems, this approach can exploit in step (2) all that is known about the equivalence of schemas ($\mathcal{D}_{can}$ and $\mathcal{D}$ in our case) expressed in the same formalism.

References

[1] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, N. Tang, The data civilizer system, in: CIDR, 2017.

[2] N. Heudecker, A. White, The data lake fallacy: All water and little substance, Gartner Report G 264950 (2014).

[3] I. Terrizzano, P. M. Schwarz, M. Roth, J. E. Colino, Data wrangling: The challenging journey from the wild to the lake, in: CIDR, 2015.

[4] P. Ciaccia, D. Martinenghi, R. Torlone, Foundations of context-aware preference propagation, J. ACM 67 (2020) 4:1–4:43. URL: https://doi.org/10.1145/3375713. doi:10.1145/3375713.

[5] CKAN: The open source data portal software, http://ckan.org/, (accessed November, 2017).

[6] A. P. Bhardwaj, A. Deshpande, A. J. Elmore, D. R. Karger, S. Madden, A. G. Parameswaran, H. Subramanyam, E. Wu, R. Zhang, Collaborative data analytics with DataHub, PVLDB 8 (2015) 1916–1927.

[7] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, S. E. Whang, Goods: Organizing google's datasets, in: SIGMOD, 2016.

[8] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, M. Donsky, G. Fierro, C. She, C. Steinbach, V. Subramanian, E. Sun, Ground: A data context service, in: CIDR, 2017.

[9] R. Hai, S. Geisler, C. Quix, Constance: An intelligent data lake system, in: F. Özcan, G. Koutrika, S. Madden (Eds.), Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, ACM, 2016, pp. 2097–2100. URL: https://doi.org/10.1145/2882903.2899389. doi:10.1145/2882903.2899389.

[10] P. Ciaccia, D. Martinenghi, R. Torlone, Preference queries over taxonomic domains, Proc. VLDB Endow. 14 (2021) 1859–1871. URL: http://www.vldb.org/pvldb/vol14/p1859-martinenghi.pdf. doi:10.14778/3467861.3467874.

[11] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, F. Naumann, Data profiling with metanome, PVLDB 8 (2015).

[12] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, Apache spark: a unified engine for big data processing, Commun. ACM 59 (2016).

[13] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, P. C. Arocena, Data lake management: Challenges and opportunities 12 (2019) 1986–1989. URL: https://doi.org/10.14778/3352063.3352116. doi:10.14778/3352063.3352116.

[14] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, S. E. Whang, Managing google's data lake: an overview of the goods system, IEEE Data Eng. Bull. 39 (2016) 5–14.

[15] Data fabric architecture is key to modernizing data management and integration, 2021. URL: https://www.gartner.com/smarterwithgartner/data-fabric-architecture-is-key-to-modernizing-data-management-and-integration.

[16] P. Ciaccia, D. Martinenghi, R. Torlone, Conceptual constraints for data quality in data lakes, in: Proceedings of the 1st Italian Conference on Big Data and Data Science (ITADATA 2022), 2022, pp. 111–122. URL: https://ceur-ws.org/Vol-3340/paper34.pdf.

[17] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, I. F. Ilyas, Guided data repair, Proc. VLDB Endow. 4 (2011) 279–289. URL: https://doi.org/10.14778/1952376.1952378. doi:10.14778/1952376.1952378.

[18] F. Chiang, R. J. Miller, A unified model for data and constraint repair, in: S. Abiteboul, K. Böhm, C. Koch, K. Tan (Eds.), Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany, IEEE Computer Society, 2011, pp. 446–457. URL: https://doi.org/10.1109/ICDE.2011.5767833. doi:10.1109/ICDE.2011.5767833.

[19] F. Geerts, G. Mecca, P. Papotti, D. Santoro, Cleaning data with llunatic, VLDB J. 29 (2020) 867–892. URL: https://doi.org/10.1007/s00778-019-00586-5. doi:10.1007/s00778-019-00586-5.

[20] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, c. Demiralp, C. Hidalgo, Sherlock: A deep learning approach to semantic data type detection, in: Proceedings of the 25th ACM SIGKDD, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1500–1508. URL: https://doi.org/10.1145/3292500.3330993. doi:10.1145/3292500.3330993.

[21] M. Ota, H. Müller, J. Freire, D. Srivastava, Data-driven domain discovery for structured datasets, Proc. VLDB Endow. 13 (2020) 953–967. URL: https://doi.org/10.14778/3384345.3384346. doi:10.14778/3384345.3384346.

[22] D. Zhang, M. Hulsebos, Y. Suhara, c. Demiralp, J. Li, W.-C. Tan, Sato: Contextual semantic type detection in tables, Proc. VLDB Endow. 13 (2020) 1835–1848. URL: https://doi.org/10.14778/3407790.3407793. doi:10.14778/3407790.3407793.

[23] X. Chu, I. F. Ilyas, S. Krishnan, J. Wang, Data cleaning: Overview and emerging challenges, in: F. Özcan, G. Koutrika, S. Madden (Eds.), Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, ACM, 2016, pp. 2201–2206. URL: https://doi.org/10.1145/2882903.2912574. doi:10.1145/2882903.2912574.

[24] M. H. Farid, A. Roatis, I. F. Ilyas, H. Hoffmann, X. Chu, CLAMS: bringing quality to data lakes, in: F. Özcan, G. Koutrika, S. Madden (Eds.), Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, ACM, 2016, pp. 2089–2092. URL: https://doi.org/10.1145/2882903.2899391. doi:10.1145/2882903.2899391.

[25] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 5511–5519. URL: https://doi.org/10.24963/ijcai.2018/777. doi:10.24963/ijcai.2018/777.

[26] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Bießmann, A. Grafberger, Automating large-scale data quality verification, Proc. VLDB Endow. 11 (2018) 1781–1794. URL: http://www.vldb.org/pvldb/vol11/p1781-schelter.pdf. doi:10.14778/3229863.3229867.

[27] S. Schelter, F. Bießmann, D. Lange, T. Rukat, P. Schmidt, S. Seufert, P. Brunelle, A. Taptunov, Unit testing data with deequ, in: P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, T. Kraska (Eds.), Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019,

ACM, 2019, pp. 1993–1996. URL: https://doi.org/10.1145/3299869.3320210. doi:10.1145/3299869.3320210.

[28] V. M. Markowitz, A. Shoshani, Representing extended entity-relationship structures in relational databases: A modular approach, ACM Trans. Database Syst. 17 (1992) 423–464. URL: https://doi.org/10.1145/132271.132273. doi:10.1145/132271.132273.