Datenbank Spektrum (2022) 22:59–66
https://doi.org/10.1007/s13222-021-00399-3

FACHBEITRAG

Four Generations in Data Engineering for Data Science

The Past, Presence and Future of a Field of Science

Meike Klettke[1] · **Uta Störl**[2]

Received: 29 May 2021 / Accepted: 22 November 2021 / Published online: 22 December 2021

Abstract
Data-driven methods and data science are important scientific methods in many research fields. All data science approaches require professional data engineering components. At the moment, computer science experts are needed for solving these data engineering tasks. Simultaneously, scientists from many fields (like natural sciences, medicine, environmental sciences, and engineering) want to analyse their data autonomously. The arising task for data engineering is the development of tools that can support an automated data curation and are utilisable for domain experts. In this article, we will introduce four generations of data engineering approaches classifying the data engineering technologies of the past and presence. We will show which data engineering tools are needed for the scientific landscape of the next decade.

Keywords Data cleaning · Data integration · Data engineering pipelines · Data curation

1 Introduction

"Drowning in Data, Dying of Thirst for Knowledge" This often used quote describes the main problems of data science: the necessity to draw useful knowledge from data and simultaneously the main aim of the data engineering field: providing data for analysis. In these dedicated application fields different kinds of data are collected and generated that shall be analysed with *data mining methods*. In this article, we use the term *data mining* in the broad interpretation synonymous to *knowledge discovery in databases* which is "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns or relationships within a dataset in order to make important decisions" (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). Even though in recent times the focus has been on artificial neural network algorithms, the entire range of data mining methods also includes clustering, classification, regression, association rules and so on.

This article, however, will mainly focus on the *data preprocessing part* of data science. Data engineering components have to read the data from very large data sources in different heterogeneous data formats and integrate the data into the target data format. In this process, data are validated, cleaned, completed, aggregated, transformed and integrated. The tools for the data engineering tasks have a long tradition in the classical database research field. For more than 50 years database management systems have been used to store large amounts of structured data. Over time, these systems have been extended and redeveloped among different dimensions:

- to handle increasing volume of data,
- to be able to store data in different data models (besides the relational data model also considering graph data model, streaming data, JSON data model) and to be able to transform data between these different models,
- to consider the heterogeneity of data, and
- to treat incompleteness and vagueness of datasets.

In data science applications, an additional requirement comes up: the wish that *domain experts* will be able to analyse their own data. Under the term *democratising* of machine learning the requirement has been exposed that

✉ Meike Klettke
meike.klettke@uni-rostock.de

Uta Störl
uta.stoerl@fernuni-hagen.de

[1] University of Rostock, Rostock, Germany

[2] University of Hagen, Hagen, Germany

lowering entry barriers for domain experts analysing their own data is necessary [37].

All above enumerated dimensions have determined the data engineering research landscape. This article will introduce a systematic classification of the field.

The rest of the article is structured as follows. In Sect. 2, a classification of data engineering methods will be introduced and four generations will be defined. Each of these generations represents a very active body of research. Thus, in Sect. 3, a comprehensive outlook on open research questions in all generations is given.

2 Classification of Data Engineering Methods

In this article, available data engineering methods for data science applications will be classified. The main contribution of the article is a systematic overview of achievements in this research field till now (First, Second, and Third Generation), the open research questions in the present (mainly in the Third Generation) and the requirements that will have to be met for the future development of the area (Fourth Generation).

The term *generation* does not mean that one generation replaces the other, but that one generation is based on the previous ones. With it no valuation is implied, but rather a temporal order once the developments began. In the following this classification of data engineering approaches will be introduced.

2.1 First Generation: Data Preprocessing

Database technology and tools for providing structured data have been available for more than 50 years. The term "data engineering" came up later. It summarises methods to provide data for business intelligence, data science analysis, and machine learning algorithms – the so-called data preprocessing.

Fig. 1 visualises data engineering as part of the data science process. This follows the observation that "data preprocessing is an often neglected but major step in the data mining process" [11]. In all real data science applications, it has been considered that data engineering is the most time-consuming subtask, estimates put the percentage at 60–80% of the total effort[1]. Reasons for this are that data preprocessing starts from scratch with each new application, a high manual effort is required which explains why it is so time-consuming, expensive and error-prone. In all real applications, data preprocessing is much more complicated than expected and numerous data quality problems, exceptions and outliers can often be found in the datasets.

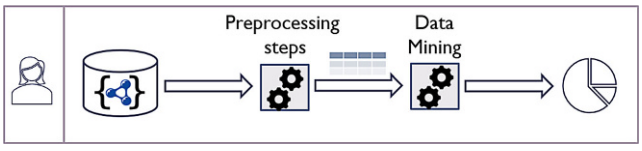

Fig. 1 First Generation: Data Engineering as part of Data Science

Because of the high amount of efforts necessary, data preprocessing has been established as its own science field and the term *data engineering* has been used for all subtasks. The high manual effort of data engineering tasks leads to the necessity of tool support. The First Generation of data engineering tools has been developed to solve different parts either to increase the data quality or to transform the data into a necessary target format. Some of the *data engineering subtasks* are:

- *Data Understanding and Data Profiling*
  - Data Exploration
  - Schema Extraction
  - Column Type Inference
  - Inference of Integrity Constraints/Pattern
- *Cleaning and Data Correction*
  - Outlier Detection and Correction
  - Duplicate Elimination
  - Missing Value Imputation
- *Data Transformation*
  - Matching and Mapping
  - Datatype Transformation
  - Transformation between different Data Models
  - Data Integration

Solutions for many data models are either based on "classic approaches" or apply machine learning algorithms to solve preprocessing tasks. In this section, an overview of some of these available approaches will be given.

There are several tutorials and textbooks that present the current state-of-the-art in the dedicated subtasks, e.g. [5, 11, 19, 29] to mention only some of these.

Data engineering of unstructured or (partially) unknown data sources often starts with *data profiling* [1]. The aim is to explore and understand the data and to derive data characteristics. Tools for data exploration give an overview of data structures, attributes, domains, regularity of data, null values, and so on, e.g. [27] for NoSQL data, in [7] a query-based approach has been suggested and in [17] an overview of available methods is given.

Schema extraction is a reverse-engineering process that extracts the implicit structural information and generates an explicit schema for a given dataset. Several algorithms

[1] "... most data scientists spend at least 80 percent of their time in data prep." [2] and "Data preparation accounts for about 80% of the work of data scientists" [32].

that deliver a schema overview have been suggested for the different data formats XML [26] and JSON [3, 23], in [34] different schema modifications for JSON data are derived (like clusters) and in [22] the complete schema history is constructed.

The reverse engineering of *column types* and the *inference of integrity constraints* like functional dependencies [4, 21] and foreign keys/inclusion dependencies [22, 24] are further subtasks in the field of data profiling.

For handling problems of low data quality, several classes of *data cleaning methods* have been developed. *Outlier detection* proves the datasets based on rules, pattern or similarity comparison and detects violations that are classified as potential data errors [6, 16, 39].

Duplicate elimination has to be applied to single data sources and also after integration of datasets from different data sources. Duplicate detection and merging of duplicate candidates based on distance functions between tuples and several methods have been developed to execute these tasks efficiently [18, 30, 31].

The *imputation of missing values* in datasets can be done with following methods: mean values or medians can be used, based on clustering the values can be estimated, blocks-wise iteration can be applied, artificial neural network algorithms and deep learning methods can also be applied to find the values.

Data transformation is another subtask of data preprocessing and realises the transformation between a source and a target structure. Each data transformation algorithm consists of *matching* source and target structures and *mapping* of the data into the target structure [8, 14, 25]. In this process *datatype transformations* can be realised. In some applications the data has to be *transformed between different data models* (e.g. NoSQL data or graph structures into relational data) and *data integration* that unifies data from different data sources in one database has to be executed. The well-studied data conflicts that have to be solved in these processes have originally been introduced in [20] and extended in [33]. Further research develops scalable data integration approaches [9].

The development of methods and implementations for the different data engineering subtasks is an ongoing task with a very active research community. Open research tasks are the adaptations of the available preprocessing methods onto new data formats, to enhance their applicability to heterogeneous data and to increase the scalability of all algorithms.

2.2 Second Generation: Data Engineering Pipelines

In the next generation of tools, the need for professionalisation of data engineering leads to tool boxes which enable the definition of data engineering pipelines that are repeatedly executed. This pipelining idea for combining data cleaning algorithms has been suggested in several publications [5, 10, 12, 13, 38]. In most tools implementing data engineering pipelines, these algorithms are applicable to different data formats, heterogeneous and distributed datasets. Thereby the diversity of input data is taken into account.

The toolboxes provide different algorithms for solving the dedicated data engineering subtasks and users have the opportunity to define processes which sequentially combine the different preprocessing algorithms. Some of these available toolsets are:

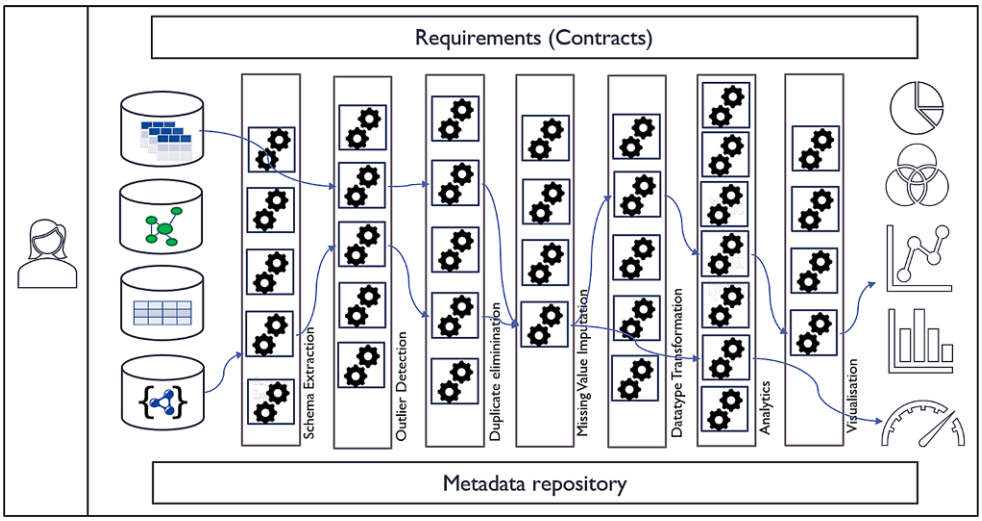

Fig. 2 Second Generation: Data Engineering/Analytics Pipelines

- ETL tools for Data Warehouses and BI tools, e.g. Talend[2], Tableau Prep[3], Qlik[4]
- Python and data science libraries, e.g. NumPy[5], pandas[6], SciPy[7], scikit-learn[8], feature-engineering[9]
- Data preparation parts in data mining tools, e.g. Weka[10], RapidMiner[11]
- Data wrangling/ Data Lake processing, e.g. Snowflake[12], IBM InfoSphere DataStage[13]

In these toolboxes, processes can be defined by composing available algorithms for continious execution. In several tools, some syntactical checks concerning the applicability of certain algorithms onto certain datasets are made (e.g. pre-test of data types and other data characteristics).

Fig. 2 visualises such toolboxes and the definition of processes (like pipelines) based on the available algorithms. It is visualised that for each data engineering subtask different algorithms are available. Their selection and combination defines the workflow for a concrete preprocessing task.

We define toolsets as Second Generation of data engineering algorithms if they are providing numerous different methods for each preprocessing subtask and for all data models and are offering the opportunity to define processes. In these toolsets the composition of the pipelines is still a manual task which is up to the user.

2.3 Third Generation: From Pipelines to Intelligent Adaptation of Data Engineering Workflows

Sect. 2.1 showed that nowadays numerous algorithms are available and ready to be used for each data engineering subtask. Each data engineering algorithm newly developed is, at the time of its publication, compared with other algorithms that exist for the same task. This is usually done on one or more datasets and should include qualitative features (like precision) and quantitative features (like efficiency).

Despite these existing comparisons, it is not easy for users of the tools to decide which algorithms in which combination are most suitable for a specific task. This re-

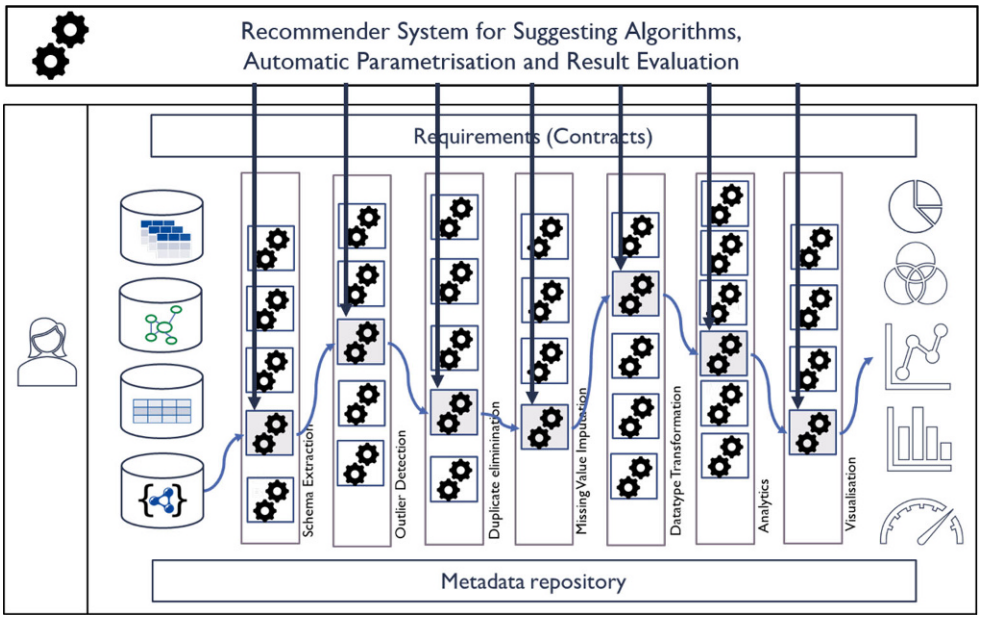

Fig. 3 Third Generation: Intelligent Advisers for Data Engineering Workflows

[2] http://www.talend.com.
[3] http://www.tableau.com/products/prep.
[4] http://www.qlik.com.
[5] http://www.numpy.org.
[6] http://www.pandas.pydata.org.
[7] http://www.scipy.org.
[8] http://www.scikit-learn.org.
[9] http://www.pypi.org/project/feature-engine.
[10] http://www.cs.waikato.ac.nz/ml/weka/.
[11] http://www.rapidminer.com.
[12] http://www.snowflake.com.
[13] http://www.ibm.com/it-infrastructure.

quires experiential knowledge and a deep understanding of all available methods and insights into the data characteristics.

This leads to an open research task: The choice of the most suitable algorithms for all subtasks and their composition has to be supported by the toolsets. Such user guidance could be provided in such a way that even if the composition task itself is up to the user, the toolset recommends applicable algorithms for each data engineering subtask, can predict expected results and can evaluate the data engineering process thus created.

The current state of the art is a bit behind this ambiguous vision. Currently, toolsets provide various implementations for all data engineering subtasks. Often they also provide the information which algorithms cannot be executed on a certain dataset, e.g. because they are not applicable to certain data formats (relation, csv, NoSQL, streaming data), or if data types (numerical values, strings, enumerations, coordinates, timestamps) do not match. The choice of the algorithms and their combination is in most cases still up to the user. As the tools claim to be usable and operable for domain experts, too, an intelligent guidance of the user, an evaluation of the results and simulation of the effects of different algorithm application are the next functionalities that the data engineering field should develop and provide.

To achieve such user guidance in workflow compositions as sketched in Fig. 3, the following building blocks are necessary:

1. Formal specification of the requirements
2. Algorithms for deriving formal metrics (e.g. schema, datatypes, pattern, constraints, data quality measures) from the datasets
3. Provision of the formal characteristics for each preprocessing algorithm in the repository of the toolset
4. Formal contracts on the pre- and postconditions for each algorithm
5. Development of a method that matches defined requirements and algorithm characteristics
6. Implementation of sample-based approaches for communication with the domain experts to explain preprocessing results
7. Evaluation of the results

This long enumeration shows that there is the need for further developments in this field at present and in the future, and that the data engineering research community is in demand here.

One very promising approach that could open an additional research direction in data engineering is currently under development in machine learning: care labels or consumer labels for machine learning algorithms [28, 36]. Comparable to care labels for textiles or description of technical devices which provide instructions on how to care or clean textiles (or how to use machine learning algorithms). The basic idea is adding metadata which rate the characteristics of certain ML algorithms. These labels would, for instance, provide information on robustness, generalisation, fairness, accuracy, and privacy sensitivity. Currently, their focus is on the analysis algorithms. Their extension to data engineering algorithms would be helpful to support the user guidance in the complete data science process orchestration and would be a building block to fulfil requirement 3 in the above enumeration. Another similar technology that could be adapted for these tasks is the formal description method for web services that have a similar aim.

2.4 Fourth Generation: Automatic Data Curation

After this already highly ambitious Third Generation, the question arises as to which further future challenges exist in data engineering research.

Currently, the available data engineering simplifies many routine tasks and avoids programming effort for the preprocessing tasks. Thus, these tools deliver a comfortable support for computer science experts. But in many application fields, domain experts have to solve the data engineering tasks. For them, the same tools are not that easy to use. There are different approaches how to overcome this problem:

- Interdisciplinary teams in Data Science projects
- Professionals who are trained in certain application fields and computer science (the development of data science master courses has this aim)
- Educational tasks for universities, teaching computer science in all university programs (e.g. natural sciences, engineering, humanities, environmental sciences, medicine)
- Development of tools for automatic data curation

Whereas the first solutions generate requirements to be met by university teaching programs, we now concentrate on the last solution: automatic data curation and want to define necessities to allow domain experts to use data curation tools and enable them to solve data persistence and usage tasks.

To approach this, let us first look at the tasks performed by a human computer science specialist in charge of data engineering in any scientific field. To define this, we first look at the tasks of *curation* in other fields such as art which is defined as: “The action or process of selecting, organising, and looking after the items in a collection or exhibition” (Oxford dictionary).

If we try to adapt this concept to *data curation* we define this item as: “Data curation is the task of controlling which data is collected, generated, captured or selected, how it is

completed, corrected and cleaned, in which schema, data format and system it is stored and how it is made available for evaluations and analytics in the long term."

Automatic data curation describes the aim to automate part of the data curation process and develop tools which either execute a certain subtask fully automated or generate recommendations and guide domain experts' decisions (semi-automatic approach).

The following vision has to be realised: The input data are datasets from a certain application that the domain experts either have created or that are the result of scientific experiments. An intelligent data curation toolset solves the following subtasks:

1. Analysis of the entire data
2. Provides information about available *standard formats* and *standard metadata formats* in this specific field of science and based on this suggests a target data format how to store or archive the data
3. Checks of the data quality
4. Intelligent guidance to clean data
5. Suggests additional data sources to complete data
6. Transforms the data into the target format and
7. Extracts the metadata for catalogues

The main difference to the Third Generation is that users need not define the target data structure in advance, as this guidance is also part of the data curation tool. This process is shown in Fig. 4. Input information is a dataset (on the left-hand side) and information about available schemas/standards in an application domain (on the right-hand side in Fig. 4). Based on this, the selection of the target format and guidance for the data engineering subtasks (cleaning and transformation) is provided. The choice of the target format can be based on calculated distances between the input datasets and the set of available standards in the dedicated science field. For this, matching algorithms [14, 33] from data integration can be applied.

The aim is to provide as much guidance as possible, supporting the choice of the target format and each data preprocessing step by recommender functions. The communication with the domain experts has to be done at each point in time with a sample-based approach, an intuitively visualisation or (pseudo-)natural language dialogue.

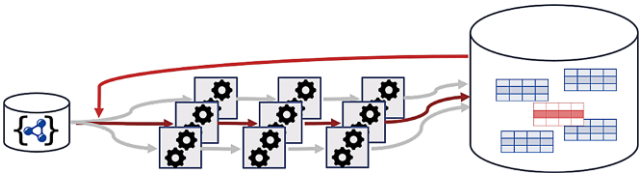

Fig. 4 Fourth Generation: Automatic Data Curation

Development of such tools for automatic data curation is an ongoing demanding task and future work for our community. The aim is to develop data engineering tools for domain scientists that are as easy to use and as intuitive as apps to provide content in social networks or WYSIWYG-Website editors.

3 Conclusion and Future Tasks

With this bold attempt to classify an entire field of science, we want to make the current and future development goals clear. The different generations of methods neither represent a chronological classification nor a valuation of the quality of the individual works. For example, there are currently high-quality works that focus on the solution of a single subtask in data engineering which achieve excellent results. In this classification, these research results would be assigned to the First Generation because they make significant scientific contributions with the development of a dedicated algorithm. Fig. 5 gives a very abstract visualisation on the relationships between the different generations.

The First Generation includes all approaches that develop a solution to a concrete data engineering task (these are several independent fields with a partial overlap, e.g. the calculation of distance functions is part of several approaches). The Second Generation represents the sequential connection of these algorithms into pipelines. In the Third Generation user guidance to compose workflows from the individual algorithms is added and in the Fourth Generation we have presented the notion of extensive support in data curation.

In each of these classes there are many open questions that represent the research tasks of the future. The main directions of this further research are:

- Optimisation of each algorithm for a dedicated data engineering subtask
- Providing implementations that are applicable for non computer-scientists out-of-the-box
- Evaluating the results of the data engineering processes (including data lineage approaches)
- Tight coupling between data engineering algorithms, machine learning implementations and result visualisation

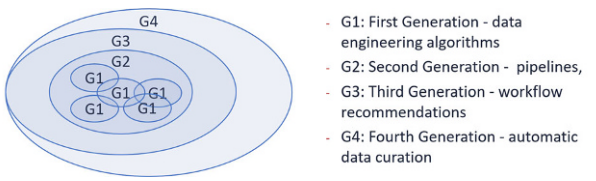

Fig. 5 Interconnection between the Four Generations of Data Engineering Approaches

methods and the joint development of cross-cutting techniques
- Development of toolsets that can provide several available data engineering algorithms and that can also be used by application experts
- *By-example* approaches for communication with domain experts, comparable to query-by-example approaches for relational databases [40]
- All four generations face significant technical challenges to maintain and evolve systems [35] and to manage evolving data [15] which are also a task for future developments.

In summary, the field of data engineering has ambitious goals for the development of further methods and tools that require a sound theoretical basis in computer science. Future development should also be increasingly interdisciplinary so that the results can be applied to all data-driven sciences.

At the same time, there is the major task of teaching computer science topics like data engineering, data literacy, machine learning, and data analytics in university education to reach future application experts in these application domains.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Abedjan Z, Golab L, Naumann F, Papenbrock T (2018) Data profiling. Synthesis lectures on data management. Morgan & Claypool Publishers,
2. Analytics India Magazine (2017) Interview with Michael Stonebraker. https://analyticsindiamag.com/interview-michael-stonebraker-distinguished-scientist-recipient-2014-acm-turing-award. Accessed: 18 Dec 2021
3. Baazizi MA, Colazzo D, Ghelli G, Sartiani C (2019) Parametric schema inference for massive JSON Datasets. VLDB J 28(4):497–521
4. Bleifuß T, Bülow S, Frohnhofen J, Risch J, Wiese G, Kruse S, Papenbrock T, Naumann F (2016) Approximate discovery of functional dependencies for large datasets. In: CIKM
5. Boehm M, Kumar A, Yang J (2019) Data management in machine learning systems. Synthesis lectures on data management. Morgan & Claypool Publishers,
6. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):15:1–15:58. https://doi.org/10.1145/1541880.1541882
7. Dimitriadou K, Papaemmanouil O, Diao Y (2014) Explore-by-example: an automatic query steering framework for interactive data exploration. SIGMOD
8. Dong XL, Halevy A, Yu C (2009) Data integration with uncertainty. VLDB J 18(2):469–500
9. Dong XL, Srivastava D (2013) Big data integration. In: Proc. ICDE. IEEE
10. Furche T, Gottlob G, Libkin L, Orsi G, Paton NW (2016) Data wrangling for big data: challenges and opportunities. In: Proc. EDBT, vol 16
11. García S, Luengo J, Herrera F (2015) Data preprocessing in data mining. Intelligent systems reference library, vol 72. Springer,
12. Golshan B, Halevy AY, Mihaila GA, Tan W (2017) Data integration: after the teenage years. In: Proc. PODS. ACM
13. Grafberger S, Stoyanovich J, Schelter S (2021) Lightweight inspection of data preprocessing in native machine learning pipelines. In: Proc. CIDR
14. Halevy A, Rajaraman A, Ordille J (2006) Data integration: the teenage years. In: Proc. VLDB
15. Hillenbrand A, Levchenko M, Störl U, Scherzinger S, Klettke M (2019) Migcast: putting a price tag on data model evolution in NoSQL data stores. In: Proc. SIGMOD
16. Hodge VJ, Austin J (2004) A survey of outlier detection methodologies. Artif Intell Rev 22(2):85–126
17. Idreos S, Papaemmanouil O, Chaudhuri S (2015) Overview of data exploration techniques. In: SIGMOD
18. Ilyas IF, Chu X (2015) Trends in cleaning relational data: consistency and deduplication. Found Trends Databases 5(4):281–393
19. Inmon WH (2005) Building the data warehouse, 4th edn. Wiley,
20. Kim W, Seo J (1991) Classifying schematic and data heterogeneity in multidatabase systems. Computer 24(12):12–18
21. Klettke M (1998) Akquisition von Integritätsbedingungen in Datenbanken. Infix Verlag, St. Augustin
22. Klettke M, Awolin H, Störl U, Müller D, Scherzinger S (2017) Uncovering the evolution history of data lakes. In: Proc. SCDM@IEEE BigData
23. Klettke M, Störl U, Scherzinger S (2015) Schema extraction and structural outlier detection for JSON-based NoSQL data stores. In: Proc. BTW
24. Kruse S, Papenbrock T, Dullweber C, Finke M, Hegner M, Zabel M, Zöllner C, Naumann F (2017) Fast approximate discovery of inclusion dependencies. In: BTW
25. Lenzerini M (2002) Data integration: a theoretical perspective. In: Proc. PODS
26. Moh C, Lim E, Ng WK (2000) DTD-miner: a tool for mining DTD from XML documents. In: Proc. WECWIS
27. Möller ML, Berton N, Klettke M, Scherzinger S, Störl U (2019) jHound: large-scale profiling of open JSON data. In: Proc. BTW
28. Morik K, Kotthaus H, Heppe L, Heinrich D, Fischer R, Pauly A, Piatkowski N (2021) The care label concept: a certification suite for trustworthy and resource-aware machine learning. In: CoRR
29. Nargesian F, Zhu E, Miller RJ, Pu KQ, Arocena PC (2019) Data lake management: challenges and opportunities. In: Proc. VLDB Endow
30. Naumann F, Herschel M (2010) An introduction to duplicate detection. Synth Lect Data Manag. https://doi.org/10.2200/S00262ED1V01Y201003DTM003
31. Panse F (2014) Duplicate detection in probabilistic relational databases. Ph.D. thesis, Staats- und Universitätsbibliothek Hamburg Carl von Ossietzky

32. Press G (2016) Cleaning big data: Most time-consuming, least enjoyable data science task. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=5bf8476f637d. Accessed: 18 Dec 2021
33. Rahm E, Bernstein PA (2001) A survey of approaches to automatic schema matching. VLDB J 10(4):334–350
34. Ruiz DS, Morales SF, Molina JG (2015) Inferring versioned schemas from NoSQL databases and its applications. In: Proc. ER, vol 9381. Springer,
35. Sculley D, Holt G, Golovin D, Davydov E, Phillips T, Ebner D, Chaudhary V, Young M, Crespo J, Dennison D (2015) Hidden technical debt in machine learning systems. In: Advances in neural information processing systems
36. Seifert C, Scherzinger S, Wiese L (2019) Towards generating consumer labels for machine learning models. In: Proc. CogMI. IEEE
37. Shang Z, Zgraggen E, Buratti B, Kossmann F, Eichmann P, Chung Y, Binnig C, Upfal E, Kraska T (2019) Democratizing data science through interactive curation of ML pipelines. In: SIGMOD
38. Terrizzano IG, Schwarz PM, Roth M, Colino JE (2015) Data wrangling: the challenging journey from the wild to the lake. In: Proc. CIDR
39. Wang H, Bah MJ, Hammad M (2019) Progress in outlier detection techniques: a survey. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2932769
40. Zloof MM (1975) Query-by-example: the invocation and definition of tables and forms. In: VLDB