*Short Paper*

# Similarity-driven Schema Transformation for Test Data Generation

Fabian Panse
University of Hamburg, Germany
fabian.panse@uni-hamburg.de

Meike Klettke
University of Rostock, Germany
meike.klettke@uni-rostock.de

Johannes Schildgen
OTH Regensburg, Germany
johannes.schildgen@oth-regensburg.de

Wolfram Wingerath
University of Oldenburg, Germany
wolfram.wingerath@uni-oldenburg.de

## ABSTRACT

A flexible and versed generation of test data is an important aspect in benchmarking algorithms for data integration. This includes the generation of heterogeneous schemas, each representing another data source of the integration benchmark. In this paper, we present our ongoing research on a novel approach for similarity-driven generation of schemas, which takes as input an arbitrary dataset, extracts its schema, and derives a set of output schemas from it. In contrast to previous solutions, we do not focus on structural transformations of relational or XML schemas, but extend the scope to contextual transformations and NoSQL data models, where the required schema information is often only implicitly defined within the data and must first be extracted. In addition, we utilize a novel method that generates multiple schemas based on user-defined heterogeneity constraints making the generation process configurable even for non-experts.

## 1 INTRODUCTION

The number of available data sources and the need to integrate them is growing rapidly in many public, academic, and industrial sectors [15]. At the same time, the increasing diversity of the database landscape makes an accurate integration of these sources considerably more difficult. All these factors have made data integration [20, 21], and all its sub-steps, such as schema matching [4, 5], duplicate detection [17, 48, 50], and record fusion [8, 53], intensively studied research areas for decades, which still receive a lot of attention today (e.g., [11, 13, 24, 25, 34, 38, 44, 46]).

The development of novel algorithms for data integration requires a systematic and thorough evaluation of them [49]. This in turn requires test datasets that contain a ground truth. Real-world datasets with ground truths are hard to find, as this truth has to be determined very laboriously and at high costs. Therefore the use of test data generators is recommended, which allow a fast generation of different benchmarking scenarios and allow the users to create datasets according to their own needs.

An important step in generating test data for data integration is to create heterogeneous data schemas, each representing a different data source of the integration benchmark, and mappings between them. Typically, this is accomplished by transforming an input schema provided by the user. Current generators of schema-related data integration benchmarks, such as iBench [3], STBenchmark [2], or MatchBench [26], however, focus on structural transformations of relational or XML schemas explicitly defined in the given data, although such schema specifications are rarely complete or, in the case of many NoSQL datasets, missing altogether. Moreover, these tools are limited to the generation of schema pairs, despite the fact that real-world integration tasks often involve more than two data sources. Finally, their configuration requires detailed knowledge on schema-transformation operators and thus are difficult to use for non-experts.

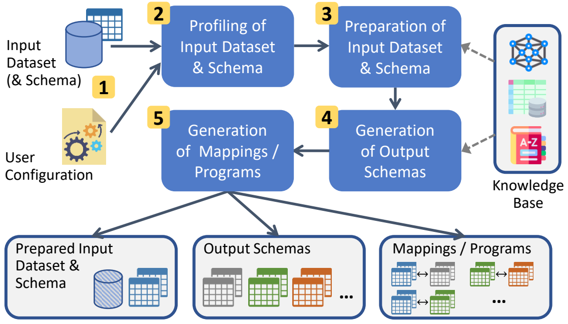

Input Dataset (& Schema)
1
User Configuration
2
Profiling of Input Dataset & Schema
3
Preparation of Input Dataset & Schema
4
Generation of Output Schemas
5
Generation of Mappings / Programs
Knowledge Base
Prepared Input Dataset & Schema
Output Schemas
...
Mappings / Programs
...

**Figure 1: Overall procedure of our approach for generating schemas for data integration benchmarks**

In this paper, we propose a novel approach to generate data schemas (and mappings between them) for data integration benchmarks that in contrast to existing solutions

- also supports NoSQL data models, such as JSON or property graphs,
- profiles the input data to enrich explicit and extract implicit schema information,
- supports contextual schema transformations, such as changing a column's format or unit of measurement, and
- utilizes a novel concept of similarity-based transformation trees to build an arbitrary number of output schemas that satisfy user-defined constraints on their heterogeneity.

The basic idea of our approach is illustrated in Figure 1. First, the user submits an arbitrary dataset (e.g., relational, JSON, or graph-based) as input along with its explicit schema (if available) and a configuration that specifies the desired heterogeneity of the output schemas to be generated. Second, the submitted dataset is profiled to identify, extract, and add missing schema information. Third, to simplify the subsequent generation of output schemas, the profiling results are used to decompose the input dataset and schema as much as possible. Fourth, the desired number of $n$ output schemas is generated by transforming the prepared input schema. To meet the user's specifications on the output schemas' heterogeneity as well as possible, we utilize a novel concept using similarity-based *transformation trees*. Finally, for each pair of schemas, two schema mappings as well as two transformation programs are generated, which will allow us later on to rewrite queries and transform data from one schema into the other. The

Series ISSN: 2367-2005

10.48786/edbt.2022.31

final output of our generation approach contains (i) the prepared input dataset and schema, (ii) $n$ output schemas, and (iii) $n(n+1)$ schema mappings and transformation programs between the individual schemas (input and output).

We plan to embed this schema generation approach into our DaPo generator [29], where we use the generated schemas to create benchmarks for duplicate detection and record fusion that consists of multiple data sources. However, the generated schemas, mappings, and programs can also be used to create benchmarks for other data integration tasks, such as schema matching/mapping [9, 34, 46], query rewriting [27], or data exchange [10].

The rest of the paper is structured as follows: First we elaborate on related work and open challenges in Section 2. Then we describe how we address schema profiling, preparation, and transformation in our research project in Sections 3 and 4. Thereafter, we discuss the calculation of schematic heterogeneities in Section 5 and present our overall generation approach in Section 6. We conclude this paper in Section 7.

## 2 RELATED WORK & OPEN CHALLENGES

Static benchmarks and experiment suites, such as Alaska [18], XBenchMatch [22], T2D [55], or Valentine [37], provide valuable test scenarios for different schema matching use cases. However, they are limited to particular domains (e.g., Alaska includes data on electronic devices) and can only be customized by removing individual schema components (e.g., sources, tables, or columns) from the test input. In addition, we cannot use them for DaPo, because it is supposed to work on arbitrary datasets.

The schema generators STBenchmark [2], MatchBench [26], and iBench [3] all support the renaming of labels. While iBench and MatchBench are limited to relational schemas, STBenchmark also provides operators for transforming nested XML schemas. The transformation of constraints is partially covered by STBenchmark (referential) and iBench (unique, referential, and functional dependencies). All three generate benchmarking scenarios consisting of one source and one target schema. Thus, it is difficult to achieve a predefined degree of heterogeneity between multiple output schemas. EMBench$^{++}$ [32] is a tool for generating entity matching benchmarks, which is able to add and remove individual columns from the benchmark's schema, but does not support more complex schema transformations.

Besides benchmarking, there is a lot of work on schema and data transformation in many other research areas. Examples are data cleaning [54], schema evolution [19, 28, 36, 61], multidatabase languages, such as SchemaSQL [39], or polyglot data management [16, 56]. Our goal is to reuse the results (e.g., transformation operators for NoSQL data models) obtained in these works whenever possible and to extend them if necessary. Despite the large amount of existing research on schema transformation, the following challenges remain for our project:

- Extraction of missing (implicit) schema information of a given dataset. This may be the whole schema if the data is managed by a schemaless NoSQL data store.
- Identification of appropriate operators for schema transformation and dependencies between them.
- Identification and collection of knowledge (e.g., ontologies) required for some schema-transformation operators.
- Developing methods to measure the heterogeneity between two schemas.
- Developing an algorithm for generating multiple schemas while considering user-defined heterogeneity constraints.

## 3 SCHEMA PROFILING & PREPARATION

Every (semi-)structured dataset follows a schema, which is either explicitly managed by the underlying database system, implicitly defined by the data-processing applications, or a mix of both (e.g, an SQL database manages the datas' structure, but the semantics of some columns is only known to the applications). Due to evolving applications, in the latter two cases, different records of the same dataset may also conform to different schema versions [58].

### 3.1 Data Schema & Categories

In the literature, data schemas are often limited to structural descriptions. In this paper, we take a broader view of the term and consider the schema as the conglomerate of all information describing the actual data. We group this information into four categories: (1) structural, (2) linguistic, (3) constraint-based, and (4) contextual. *Linguistic* schema information refers to (the semantics of) labels, such as the names of relational tables and columns, XML tags, or field names in field-value pairs (e.g., ('name','Ian')). *Constraint-based* information refers to integrity constraints ranging from keys to application-specific conditions. *Contextual* information encompasses all remaining information necessary to fully interpret individual data objects (e.g., tables, columns, or values). For example, the context of a column includes its (i) format (e.g., 'yyyy-mm-dd' vs. 'dd.mm.yy'), (ii) level of abstraction (e.g., district vs. city), (iii) unit of measurement (e.g., 'cm' vs. 'inch'), and (iv) encoding (e.g., {yes,no} vs. {1,0}). The context of a table includes its scope (e.g., 'book' vs. 'novel').

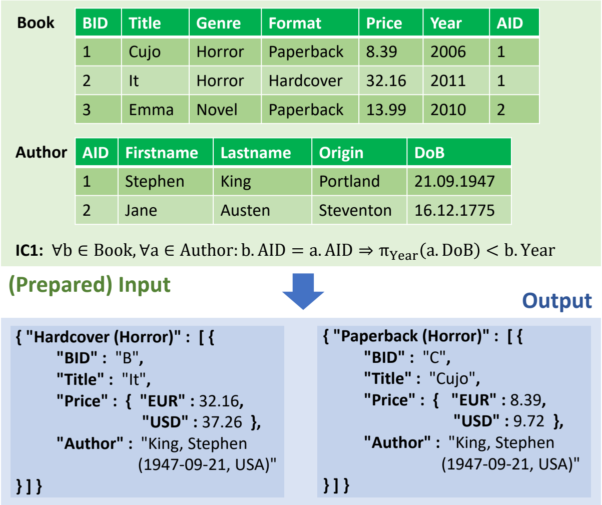
### 3.2 Data & Schema Profiling

Many datasets do not contain an explicit description of their complete schema. However, the more detailed schema information we have, the greater the choice of transformation operators we can apply to it. Thus, it is important to derive a schema from the input data that is as accurate, complete, and detailed as possible.

The profiling of data is currently a hot topic in the database community [1], and there is already a lot of (ongoing) research on identifying and extracting (i) schema information in/from CSV files [33], JSON documents [35], and graph databases [40], (ii) integrity constraints, such as unique [7], denial constraints [45, 52], inclusion dependencies [59], or functional dependencies [6, 14, 51, 57], and (iii) semantic domains [31, 62], that we reuse for our purpose. However, the identification of some contextual information, such as the scope of a table or the unit of measurement of a column, has not yet received much attention and needs further research. The same applies to identifying the semantic closeness of columns to determine which of them are likely to merge.

### 3.3 Data & Schema Preparation

After we have profiled the input data, the obtained schema information is used to further decompose the input dataset and schema so that their information is represented in as much detail as possible. This decomposition has the goal to simplify subsequent transformations. For example, it is easier to merge two attributes than to split one. Therefore, we transform the input dataset into a structured data model, normalize its schema, and split its attributes into several subattributes if a clear separation between the corresponding values is possible. Moreover, if its records conform to different schema versions, they are all initially migrated to the same version (e.g., the latest one) [36].



Book

| BID | Title | Genre | Format | Price | Year | AID |
|---|---|---|---|---|---|---|
| 1 | Cujo | Horror | Paperback | 8.39 | 2006 | 1 |
| 2 | It | Horror | Hardcover | 32.16 | 2011 | 1 |
| 3 | Emma | Novel | Paperback | 13.99 | 2010 | 2 |

Author

| AID | Firstname | Lastname | Origin | DoB |
|---|---|---|---|---|
| 1 | Stephen | King | Portland | 21.09.1947 |
| 2 | Jane | Austen | Steventon | 16.12.1775 |

**IC1:** $\forall b \in \mathrm{Book}, \forall a \in \mathrm{Author}: b.\mathrm{AID} = a.\mathrm{AID} \Rightarrow \pi_{\mathrm{Year}}(a.\mathrm{DoB}) < b.\mathrm{Year}$

```
{ "Hardcover (Horror)" :  [ {
        "BID" :  "B",
        "Title" :  "It",
        "Price" :  {  "EUR" : 32.16,
                      "USD" : 37.26  },
        "Author" :  "King, Stephen
                    (1947-09-21, USA)"
} ] }
```

```
{ "Paperback (Horror)" :  [ {
        "BID" :  "C",
        "Title" :  "Cujo",
        "Price" :  {   "EUR" : 8.39,
                       "USD" : 9.72  },
        "Author" :  "King, Stephen
                    (1947-09-21, USA)"
} ] }
```

**Figure 2: Example of Schema & Data Transformations**

## 4 SCHEMA-TRANSFORMATION OPERATORS

According to the four categories of schema information presented in Section 3.1, we distinguish the same four categories of schema-transformation operators. Since we decomposed and detailed the input schema in the preparation step as much as possible, we do not need structural operators that decompose a schema further, such as normalize a table or split a column, and do not need contextual operators that increase the level of detail, such as drill-down or an increase of precision. Nevertheless, it should be noted that there are still operators which decompose existing schema elements if this is part of a restructuring process, such as (un)nesting or regrouping (see example described below).

Constraint-based operators change the set of given integrity constraints. This can be the addition of a new constraint or the removal, strengthening or weakening of an existing constraint. Obviously if we just migrate the data of our input instance to these output schemas, every removed constraint will still be satisfied. However a removal of those constraints is relevant, if the data is further polluted with data errors as in DaPo.

Operators of all four categories are illustrated in Figure 2. We have several structural transformations. For instance, the *Book* and the *Author* tables are joined, restructured by grouping all records based on their *formats*, and then converted into JSON collections. Moreover, the columns *Firstname*, *Lastname*, *DoB*, and *Origin* are merged into the property *Author*. Finally, the *book price* is added in dollar and both price values are nested into one property *Price*. In addition, we have several contextual transformations. First, the abstraction level of the column *Origin* was drilled-up from city to country. Second, the format of the column *DoB* was changed. Third, the scope of the *Book* table was reduced to the *genre* 'horror'. Examples for linguistic transformations are the renaming of the two *Book* collections and the two *Price* properties. The only constraint-based transformation is the removal of integrity constraint *IC1* which was necessary because the *Year* values were removed from the two book collections.

### 4.1 Transformation Dependencies

It is important to note that the execution of one operator may require the subsequent execution of others. For example, if we merge two columns, we need to define a new column name. The same may apply if we increase the level of abstraction of a column (drill-up). Thus, we have dependencies between the four aforementioned categories. Typically, a structural operator implies a linguistic or contextual operator, and a contextual operator implies a linguistic one, but not vice versa. In addition, changing a context may require to change an integrity constraint. For example, when converting the unit of measurement of a column from 'feet' to 'cm', we may need to adapt a constraint that restricts the maximum size value. Finally, linguistic transformations also often require a refactoring of constraints. This leads us to the following (approximate) dependency order:

$$\textit{structural} \longrightarrow \textit{contextual} \longrightarrow \textit{linguistic} \longrightarrow \textit{constraint} \qquad (1)$$

### 4.2 Required Knowledge

Several transformation operators require additional information, which we store in a knowledge base (see Figure 1). Most structural transformations only require knowledge about the data model with which the given schema is defined. It becomes more complex if the schema has to be transformed from one model (e.g., relational) into another (e.g., JSON). In this case, we need transformation rules, either directly between both models (e.g., [41, 56]) or indirectly via a third model, which can be a generic one such as U-schema [12].

In addition to these mappings, we need dictionaries and ontologies (e.g., from DBpedia [43]) to enable linguistic and contextual transformations addressing semantic relations, such as synonyms or hyperonyms. For changing units of measurement, we need conversion rules, which in turn may be time-variant (e.g., the daily changing exchange rate between two currencies). Finally, changing the format or encoding of a column requires alternative (and common) representations and terminologies of the corresponding domain, which we collect from other datasets, such as the Dresden Web Tables Corpus [23] or GitTables [30].

## 5 HETEROGENEITY CALCULATION

Systematic benchmarking requires that the user is able to generate test datasets with varying degrees of heterogeneity. To assist inexperienced users who are unable to map such a degree to a corresponding sequence of transformation operators, we need measures to quantify the heterogeneity between the generated output schemas. Since heterogeneity can be seen as the conceptual opposite of similarity, we can use common similarity measures for this purpose. Such measures can greatly differ from one schema category to another. For this reason, we separate our measurement into four parts accordingly and model the heterogeneity of two schemas by a quadruple $h \in [0, 1]^4$ where each of the tuple's values represents the normalized heterogeneity with respect to one of the four schema categories (see Section 3.1).

Subsequent calculations with those quadruples follow the rules of component-wise addition and scalar multiplication, i.e., let $v, w \in \mathbb{R}^4$ and $\lambda \in \mathbb{R}$, it holds:

$$\pi_k(v + w) = \pi_k(v) + \pi_k(w) \qquad (2)$$

$$\pi_k(\lambda \cdot v) = \lambda \cdot \pi_k(v) \qquad (3)$$

where $\pi_k(v)$ gives the $k$th entry of tuple $v$. Moreover, it holds:

$$\pi_k(op(v, w)) = op(\pi_k(v), \pi_k(w)) \qquad (4)$$

for $op \in \{min, max\}$.

The meaning of structural similarity between two schemas strongly depends on the available structures and thus can greatly differ between the individual data models. Existing measures

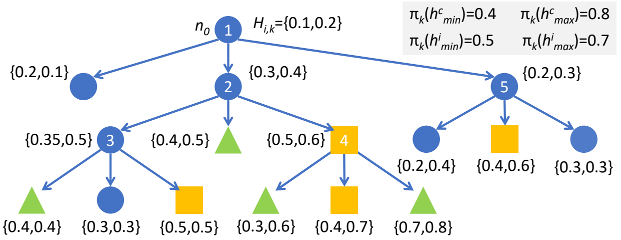


**Figure 3: Example of a transformation tree. The numbers indicate the order in which the nodes were expanded. Valid (▲) and target (■) nodes are colored and shaped differently.**

to capture such similarities include XClust [42] for hierarchical XML schemas and similarity flooding [47] for relational schemas.

We can use measures from string matching, such as Soundex or Levenshtein [20], to compare labels. Contexts affect the actual data. Thus, one way to compare two contexts is by comparing a small sample of duplicate records from the compared datasets. Here we can again use similarity measures for string matching.

The simplest way to compare two sets of integrity constraints, is to calculate their set-based similarity by using measures such as Jaccard or Dice [20]. In that case, however, it is lost that different constraints can be very similar in their semantics. However, we are not aware of existing measures that are able to capture such similarities. Nevertheless, Türker and Saake define various relationships between integrity constraints [60], which can be used as a starting point to develop such a measure.

## 6 GENERATION PROCESS

The input to a generation task are (i) the prepared input dataset and schema, and (ii) a user configuration including $n$, the number of output schemas to be generated. In the configuration, the user can define which transformation operators may be used during the generation process. The most important parameters, however, are the three quadruples $h^c_{min}, h^c_{max}, h^c_{avg} \in [0,1]^4$ that allow the user to control the minimal, maximal, and average degree of heterogeneity between the generated schemas. Obviously, it has to hold $\pi_k(h^c_{min}) \le \pi_k(h^c_{avg}) \le \pi_k(h^c_{max})$ for $k \in \{1,\ldots,4\}$.

Based on this information, the output schemas $\mathcal{S} = \{S_1,\ldots,S_n\}$ are generated one after another, each by transforming the prepared input schema, so that $\forall k \in \{1,\ldots,4\}$:

$$\forall S_i, S_j \in \mathcal{S}, i \neq j\colon \pi_k(h(S_i,S_j)) \in [\pi_k(h^c_{min}), \pi_k(h^c_{max})] \quad (5)$$

$$\sum\nolimits_{S_i,S_j \in \mathcal{S}, i\neq j} \pi_k(h(S_i,S_j)) \approx \pi_k(h^c_{avg})\frac{n(n-1)}{2} \quad (6)$$

### 6.1 Overall Generation Procedure

To meet the parameter $h^c_{avg}$ for the average heterogeneity of the schemas as closely as possible (Equation 6), new threshold values are determined at the beginning of every run $i \in \{1,\ldots,n\}$. Logically, each newly generated schema can only be compared to the previously generated schemas. This leads to the fact that the number of newly added pairwise heterogeneity tuples $h \in [0,1]^4$ per run increases as the generation task progresses. While the first run does not yet generate any of these tuples, the fourth run already generates three. Consequently, the last run has the highest impact on the final average. We have to consider this imbalance during the calculation of the aforementioned thresholds. Let $\rho_i$ be the number of pairwise schema comparisons that remain before starting run $i$. It is initially set to $\rho_1 = n(n-1)/2$ and decreases by $i-1$ after the $i$th run, i.e., $\rho_{i+1} = \rho_i - (i-1)$. Moreover let $\sigma_i \in \mathbb{R}^4$ be the total sum of heterogeneity we still need at the start of run $i$ to meet the user's specification. Initially, it equals $\sigma_1 = \rho_1 \cdot h^c_{avg}$ and decreases by $h_i = \sum_{j=1}^{i-1} h(S_i,S_j)$ after we have generated schema $S_i$ in the $i$th run. i.e., $\sigma_{i+1} = \sigma_i - h_i$. Based on these numbers, the two thresholds are calculated as:

$$h^i_{min} = max(h^c_{min}, (\sigma_i - \rho_{i+1} \cdot h^c_{max})/(i-1)) \quad (7)$$

$$h^i_{max} = min(h^c_{max}, (\sigma_i - \rho_{i+1} \cdot h^c_{min})/(i-1)) \quad (8)$$

### 6.2 Generation of Each Output Schema

Due to the dependency order described in Section 4.1, the generation of each output schema $S_i$ is executed in four steps:

(1) structural transformations
(2) contextual transformations
(3) linguistic transformations
(4) constraint-based transformations

Between every two steps, dependent transformations of the following categories are identified and executed.

In each step $k \in \{1,\ldots,4\}$, we span a so-called *transformation tree* (see Figure 3). The root node $n_0$ represents the schema resulting from the previous step (or the prepared input schema if $k=1$). This node is expanded by applying a predefined number of transformations. The resulting schemas form the child nodes of $n_0$. Then, for each of these schemas $S$, the heterogeneity to all already generated output schemas is measured regarding the schema category of the respective step (i.e., structural if $k=1$). The result is the heterogeneity bag $H_{i,k}(S) = \{\pi_k(h(S,S_j)) \mid j < i\}$. The node of schema $S$ is called *valid* if

$$\forall h \in H_{i,k}(S)\colon h \in [\pi_k(h^c_{min}), \pi_k(h^c_{max})] \quad (9)$$

In addition, a valid node is called a *target* node if

$$avg(H_{i,k}(S)) \in [\pi_k(h^i_{min}), \pi_k(h^i_{max})] \quad (10)$$

Next, a leaf node of the current tree is expanded. If there is already a target node by then, the node to be expanded is selected randomly among all leaf nodes. If this is not the case, the distance to the range $[\pi_k(h^i_{min}), \pi_k(h^i_{max})]$ is calculated for all leaf nodes, and then the leaf node with the smallest distance is expanded.

The construction of the tree ends after a predefined number of nodes have been expanded. If there are target nodes by then, one of them is chosen randomly as the output schema $S_i$. Otherwise, the schema of the node with the smallest distance is returned as output where valid nodes are preferred to non-valid ones.

## 7 CONCLUSION & ONGOING WORK

In this paper, we presented a novel approach for generating a set of heterogeneous data schemas as needed in data integration benchmarks. In contrast to existing solutions, it is (i) not limited to relational or XML schemas, (ii) able to deal with implicit schema information, (iii) also considering contextual schema transformations, (iv) able to generate scenarios with more than two schemas, and (v) similarity-driven, i.e., the generation is based on user-defined similarity scores, which ease its configuration.

The research presented in this paper is ongoing work which is being implemented in the context of the DaPo project [29]. In this project, we use the resulting schemas as input to a data pollution process to generate realistic benchmarks for duplicate detection and record fusion consisting of multiple heterogeneous data sources. The next steps of the project include the development of (i) similarity measures for the different schema categories, and (ii) a filter that selects suitable transformation operators depending on the respective node of the transformation tree.

# REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling Relational Data: A Survey. *VLDB J.* 24, 4 (2015), 557–581. https://doi.org/10.1007/s00778-015-0389-y

[2] Bogdan Alexe, Wang Chiew Tan, and Yannis Velegrakis. 2008. STBenchmark: Towards a Benchmark for Mapping Systems. *Proceedings of the VLDB Endowment* 1, 1 (2008), 230–244. https://doi.org/10.14778/1453856.1453886

[3] Patricia C. Arocena, Boris Glavic, Radu Ciucanu, and Renée J. Miller. 2015. The iBench Integration Metadata Generator. *Proceedings of the VLDB Endowment* 9, 3 (2015), 108–119. https://doi.org/10.14778/2850583.2850586

[4] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm. 2011. *Schema Matching and Mapping.* Springer.

[5] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic Schema Matching, Ten Years Later. *Proceedings of the VLDB Endowment* 4, 11 (2011), 695–701. http://www.vldb.org/pvldb/vol4/p695-bernstein_madhavan_rahm.pdf

[6] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuruganathan. 2019. Discovery of Genuine Functional Dependencies from Relational Data with Missing Values. In *Actes du XXXVIIème Congrès INFORSID.* 287–288. http://inforsid.fr/actes/2019/INFORSID_2019_p287-288.pdf

[7] Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, and Martin Schirneck. 2020. Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2270–2283. http://www.vldb.org/pvldb/vol13/p2270-birnick.pdf

[8] Jens Bleiholder and Felix Naumann. 2008. Data fusion. *ACM Comput. Surv.* 41, 1 (2008).

[9] Angela Bonifati, Ugo Comignani, Emmanuel Coquery, and Romuald Thion. 2019. Interactive Mapping Specification with Exemplar Tuples. *ACM Trans. Database Syst.* 44, 3 (2019), 10:1–10:44. https://doi.org/10.1145/3321485

[10] Angela Bonifati, Ugo Comignani, and Efthymia Tsamoura. 2021. Exchanging Data under Policy Views. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT.* OpenProceedings.org, 1–12. https://doi.org/10.5441/002/edbt.2021.02

[11] Maxime Buron, François Goasdoué, Ioana Manolescu, and Marie-Laure Mugnier. 2020. Obi-Wan: Ontology-Based RDF Integration of Heterogeneous Data. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2933–2936. https://doi.org/10.14778/3415478.3415512

[12] Carlos Javier Fernández Candel, Diego Sevilla Ruiz, and Jesús Joaquín García Molina. 2021. A Unified Metamodel for NoSQL and Relational Databases. *CoRR* (2021). https://arxiv.org/abs/2105.06494

[13] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference.* ACM, 1335–1349. https://doi.org/10.1145/3318464.3389742

[14] Loredana Caruccio, Vincenzo Deufemia, Felix Naumann, and Giuseppe Polese. 2021. Discovering Relaxed Functional Dependencies Based on Multi-Attribute Dominance. *IEEE Trans. Knowl. Data Eng.* 33, 9 (2021), 3212–3228. https://doi.org/10.1109/TKDE.2020.2967722

[15] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset Search: A Survey. *VLDB J.* 29, 1 (2020), 251–272. https://doi.org/10.1007/s00778-019-00564-x

[16] Alberto Hernández Chillón, Diego Sevilla Ruiz, and Jesús García Molina. 2021. Towards a Taxonomy of Schema Changes for NoSQL Databases: The Orion Language. In *Proceedings of the 40th International Conference on Conceptual Modeling, ER*, Vol. 13011. Springer, 176–185. https://doi.org/10.1007/978-3-030-89022-3_15

[17] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer.

[18] Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, Federico Piai, and Divesh Srivastava. 2021. Alaska: A Flexible Benchmark for Data Integration Tasks. *CoRR* (2021). https://arxiv.org/abs/2101.11259

[19] Carlo Curino, Hyun Jin Moon, Alin Deutsch, and Carlo Zaniolo. 2013. Automating the Database Schema Evolution Process. *VLDB J.* 22, 1 (2013), 73–98. https://doi.org/10.1007/s00778-012-0302-x

[20] AnHai Doan, Alon Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration.* Morgan Kaufmann.

[21] Xin Luna Dong and Divesh Srivastava. 2015. *Big Data Integration.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00578ED1V01Y201404DTM040

[22] Fabien Duchateau and Zohra Bellahsene. 2014. Designing a Benchmark for the Assessment of Schema Matching Tools. *Open J. Databases* 1, 1 (2014), 3–25. https://nbn-resolving.org/urn:nbn:de:101:1-201705194573

[23] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the Dresden Web Table Corpus: A Classification Approach. In *Proceedings of the 2nd IEEE/ACM International Symposium on Big Data Computing, BDC.* IEEE Computer Society, 41–50. https://doi.org/10.1109/BDC.2015.30

[24] Martin Franke, Ziad Sehili, Florens Rohde, and Erhard Rahm. 2021. Evaluation of Hardening Techniques for Privacy-Preserving Record Linkage. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT.* 289–300. https://doi.org/10.5441/002/edbt.2021.26

[25] Leonardo Gazzarri and Melanie Herschel. 2021. End-to-end Task Based Parallelization for Entity Resolution on Dynamic Data. In *Proceedings of the 37th IEEE International Conference on Data Engineering, ICDE.* IEEE, 1248–1259. https://doi.org/10.1109/ICDE51399.2021.00112

[26] Chenjuan Guo, Cornelia Hedeler, Norman W. Paton, and Alvaro A. A. Fernandes. 2013. MatchBench: Benchmarking Schema Matching Algorithms for Schematic Correspondences. In *Proceedings of the 29th British National Conference on Databases, BNCOD*, Vol. 7968. Springer, 92–106. https://doi.org/10.1007/978-3-642-39467-6_11

[27] Alon Y. Halevy. 2001. Answering Queries using Views: A Survey. *VLDB J.* 10, 4 (2001), 270–294. https://doi.org/10.1007/s007780100054

[28] Kai Herrmann, Hannes Voigt, Jonas Rausch, Andreas Behrend, and Wolfgang Lehner. 2018. Robust and Simple Database Evolution. *Inf. Syst. Frontiers* 20, 1 (2018), 45–61. https://doi.org/10.1007/s10796-016-9730-2

[29] Kai Hildebrandt, Fabian Panse, Niklas Wilcke, and Norbert Ritter. 2020. Large-Scale Data Pollution with Apache Spark. *IEEE Trans. Big Data* 6, 2 (2020), 396–411. https://doi.org/10.1109/TBDATA.2016.2637378

[30] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2021. GitTables: A Large-Scale Corpus of Relational Tables. *CoRR* (2021). https://arxiv.org/abs/2106.07258

[31] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD.* ACM, 1500–1508. https://doi.org/10.1145/3292500.3330993

[32] Ekaterini Ioannou and Yannis Velegrakis. 2019. EMBench$^{++}$: Data for a Thorough Benchmarking of Matching-Related Methods. *Semantic Web* 10, 2 (2019), 435–450. https://doi.org/10.3233/SW-180331

[33] Lan Jiang, Gerardo Vitagliano, and Felix Naumann. 2021. Structure Detection in Verbose CSV Files. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT.* OpenProceedings.org, 193–204. https://doi.org/10.5441/002/edbt.2021.18

[34] Angelika Kimmig, Alex Memory, Renée J. Miller, and Lise Getoor. 2019. A Collective, Probabilistic Approach to Schema Mapping Using Diverse Noisy Evidence. *IEEE Trans. Knowl. Data Eng.* 31, 8 (2019), 1426–1439. https://doi.org/10.1109/TKDE.2018.2865785

[35] Meike Klettke, Uta Störl, and Stefanie Scherzinger. 2015. Schema Extraction and Structural Outlier Detection for JSON-based NoSQL Data Stores. In *BTW.* 425–444. https://dl.gi.de/20.500.12116/2420

[36] Meike Klettke, Uta Störl, Manuel Shenavai, and Stefanie Scherzinger. 2016. NoSQL Schema Evolution and Big Data Migration at Scale. In *Proceedings of the 2016 IEEE International Conference on Big Data, BigData.* IEEE Computer Society, 2764–2774. https://doi.org/10.1109/BigData.2016.7840924

[37] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *Proceedings of the 37th IEEE International Conference on Data Engineering, ICDE.* IEEE, 468–479. https://doi.org/10.1109/ICDE51399.2021.00047

[38] Shrinu Kushagra, Hemant Saxena, Ihab F. Ilyas, and Shai Ben-David. 2019. A Semi-Supervised Framework of Clustering Selection for De-Duplication. In *Proceedings of the 35th IEEE International Conference on Data Engineering, ICDE.* IEEE, 208–219. https://doi.org/10.1109/ICDE.2019.00027

[39] Laks V. S. Lakshmanan, Fereidoon Sadri, and Subbu N. Subramanian. 2001. SchemaSQL: An Extension to SQL for Multidatabase Interoperability. *ACM Trans. Database Syst.* 26, 4 (2001), 476–519.

[40] Hanâ Lbath, Angela Bonifati, and Russ Harmer. 2021. Schema Inference for Property Graphs. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT.* OpenProceedings.org, 499–504. https://doi.org/10.5441/002/edbt.2021.58

[41] Dongwon Lee, Murali Mani, and Wesley W. Chu. 2003. Schema Conversion Methods between XML and Relational Models. In *Knowledge Transformation for the Semantic Web*, Borys Omelayenko and Michel C. A. Klein (Eds.). Frontiers in Artificial Intelligence and Applications, Vol. 95. IOS Press, 1–17.

[42] Mong-Li Lee, Liang Huai Yang, Wynne Hsu, and Xia Yang. 2002. XClust: Clustering XML Schemas for Effective Integration. In *Proceedings of the 11th International Conference on Information and Knowledge Management, CIKM.* ACM, 292–299. https://doi.org/10.1145/584792.584841

[43] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195. https://doi.org/10.3233/SW-140134

[44] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60. https://doi.org/10.14778/3421424.3421431

[45] Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. 2020. Approximate Denial Constraints. *Proceedings of the VLDB Endowment* 13, 10 (2020), 1682–1695. https://doi.org/10.14778/3401960.3401966

[46] Lacramioara Mazilu, Norman W. Paton, Alvaro A. A. Fernandes, and Martin Koehler. 2019. Dynamap: Schema Mapping Generation in the Wild. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management, SSDBM.* ACM, 37–48. https://doi.org/10.1145/3335783.3335785

[47] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proceedings of the 18th IEEE International Conference on Data Engineering, ICDE.* IEEE Computer Society, 117–128. https://doi.org/10.1109/ICDE.2002.994702

[48] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00262ED1V01Y201003DTM003

[49] Fabian Panse and Felix Naumann. 2021. Evaluation of Duplicate Detection Algorithms: From Quality Measures to Test Data Generation. In *Proceedings of the 37th IEEE International Conference on Data Engineering, ICDE.* IEEE, 2373–2376. https://doi.org/10.1109/ICDE51399.2021.00269

[50] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. *The Four Generations of Entity Resolution.* Morgan & Claypool Publishers. https://doi.org/10.2200/S01067ED1V01Y202012DTM064

[51] Thorsten Papenbrock and Felix Naumann. 2016. A Hybrid Approach to Functional Dependency Discovery. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference.* ACM, 821–833. https://doi.org/10.1145/2882903.2915203

[52] Eduardo H. M. Pena, Eduardo Cunha de Almeida, and Felix Naumann. 2019. Discovery of Approximate (and Exact) Denial Constraints. *Proceedings of the VLDB Endowment* 13, 3 (2019), 266–278. https://doi.org/10.14778/3368289.3368293

[53] Romila Pradhan, Siarhei Bykau, and Sunil Prabhakar. 2017. Staging User Feedback toward Rapid Conflict Resolution in Data Fusion. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference.* ACM, 603–618. https://doi.org/10.1145/3035918.3035941

[54] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of 27th International Conference on Very Large Data Bases, VLDB.* 381–390. http://www.vldb.org/conf/2001/P381.pdf

[55] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching HTML Tables to DBpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS.* ACM, 10:1–10:6. https://doi.org/10.1145/2797115.2797118

[56] Johannes Schildgen, Yannick Krück, and Stefan Deßloch. 2017. Transformations on Graph Databases for Polyglot Persistence with NotaQL. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS) (LNI)*, Vol. P-265. GI, 83–102. https://dl.gi.de/20.500.12116/677

[57] Philipp Schirmer, Thorsten Papenbrock, Sebastian Kruse, Felix Naumann, Dennis Hempfing, Torben Mayer, and Daniel Neuschäfer-Rube. 2019. DynFD: Functional Dependency Discovery in Dynamic Datasets. In *Proceedings of the 22nd International Conference on Extending Database Technology, EDBT.* OpenProceedings.org, 253–264. https://doi.org/10.5441/002/edbt.2019.23

[58] Uta Störl, Meike Klettke, and Stefanie Scherzinger. 2020. NoSQL Schema Evolution and Data Migration: State-of-the-Art and Opportunities. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT.* OpenProceedings.org, 655–658. https://doi.org/10.5441/002/edbt.2020.87

[59] Fabian Tschirschnitz, Thorsten Papenbrock, and Felix Naumann. 2017. Detecting Inclusion Dependencies on Very Many Tables. *ACM Trans. Database Syst.* 42, 3 (2017), 18:1–18:29. https://doi.org/10.1145/3105959

[60] Can Türker and Gunter Saake. 1998. Deriving Relationships between Integrity Constraints for Schema Comparison. In *Proceedings of the 2nd East European Symposium on Advances in Databases and Information Systems, ADBIS*, Vol. 1475. Springer, 188–199. https://doi.org/10.1007/BFb0057732

[61] Yannis Velegrakis, Renée J. Miller, and Lucian Popa. 2004. Preserving mapping consistency under schema changes. *VLDB J.* 13, 3 (2004), 274–293. https://doi.org/10.1007/s00778-004-0136-2

[62] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çagatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual Semantic Type Detection in Tables. *Proceedings of the VLDB Endowment* 13, 11 (2020), 1835–1848. http://www.vldb.org/pvldb/vol13/p1835-zhang.pdf