Real-time visualization of profiling metadata upon data insertions

Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, Giuseppe Polese
Department of Computer Science
University of Salerno
Fisciano (SA), Italy
{lcaruccio,scirillo,deufemia,gpolese}@unisa.it

ABSTRACT

Data-intensive processes must deal with the problem of monitoring the quality of data. To this end, metadata can be exploited in order to highlight errors and support the cleaning of data. In this paper, we present a novel DBMS plugin for validating profiling metadata during data insertions, aiming to assist the user in a-priori checking the quality of data being inserted into a database. It has been implemented within the MySQL Workbench client and is able to intercept and validate data insertion queries. More specifically, using such a plugin it is possible to verify in real-time whether the data to be inserted into a database instance will produce some violations on specific metadata, such as unique column combinations or functional dependencies. A user study highlighted the effectiveness of the proposed plugin by also emphasizing its strengths and weaknesses.

KEYWORDS

Data profiling, Functional dependencies, Data insertions, MySQL workbench

1 INTRODUCTION

Data quality is becoming a vital activity in most application domains. The presence of errors and inconsistencies in the data drastically reduce the value of data and processing capabilities on them [13]. Nowadays, data-intensive systems are involved in everyday life and enable complex analysis based on the data they process. Examples are data warehousing and data mining systems, through which statistics on data, classification tasks, and in general novel insights can be generated. Data is a valuable asset for customers, companies, and governments, which yields their quality to have a fundamental role in data governance models [12]. Nevertheless, the possible presence of dirty data might produce negative effects on their results. To this end, big efforts have been devoted to cleaning data before employing them in data-intensive processes. In fact, many approaches to automatically perform cleaning tasks, such as de-duplication, missing value imputation, and entity resolution represent a landmark for guaranteeing a proper drawing of the reality through the data describing it. Such approaches permit to automate and/or optimize data preparation tasks at semantic-level, which would otherwise require bigger efforts, tedious and long lasting pre-processing activities. In fact, it is well-known that data scientists spend most of their time in data preparation tasks [18].

Many data cleaning approaches exploit *profiling metadata*, that permit to describe properties that must hold on data, and consequently they can be used to monitor the quality of the data themselves. Examples of metadata are unique column combinations, functional dependencies, denial constraints, and so on. Moreover, nowadays the exploitation and validation of these metadata are enhanced thanks to recent data profiling algorithms, which enable their automatic discovery from "big" datasets [1, 4, 10, 21].

DataBase Management Systems (DBMSs) have been recognized as a useful tool enabling easy interactions between the user and the stored data. In particular, they permit to inquire data with little effort [2]. They can also be used to perform data fusion activities, after schema matching processes [5]. In such scenarios, it would be desirable that the validation of profiling metadata be managed by users through the definition of some constraints. However, to the best of our knowledge, DBMSs do not permit to verify the validity of metadata, such as FD's satisfiability, on the current database instance, and/or verify whether some of them could be invalidated upon data insertion operations.

In this paper we introduce a novel visual tool named INDITIO (INteracting with metaData during data InserTIOns), which permits to monitor in real-time, and possibly validate, specific profiling metadata upon data insertion operations. In particular, the tool has been implemented as a plugin of the graphical client MySQL Workbench, and it enables the user to intercept data insertion queries, in order to validate specified (or uploaded) metadata before the insertion is committed. In this way, a user can check the correctness of the data s/he is planning to insert, and decide either to force the insertion nonetheless the data, or try to fix them. Consequently, INDITIO manages the specification (uploading) of metadata and the visualization of validation results together with values inducing possible violations. Moreover, different statistical counters and visual components enable users to have an overview of the general impact of data insertions on the considered metadata. The current implementation of the plugin is able to manage unique column combinations (UCCs) and functional dependencies (FDs), but it could be extended to include other kinds of profiling metadata.

In general, the goal that guided such work is threefold: i) improve the quality of data by driving users in the data insertion process, ii) sensitize more users to the data profiling concepts, and iii) support an implicit analysis on the significance of metadata from domain experts.

The paper is organized as follows. Section 2 describes approaches to data quality and data profiling tools. Section 3 presents the theoretical foundations of considered profiling metadata. Section 4 presents INDITIO, whereas Section 5 reports a user study we performed to analyze its effectiveness and usability. Finally, summary and future directions are included in Section 6.

2 RELATED WORK

Metadata have been recognized as a fundamental tool for guaranteeing good levels of quality to the extension of data, mainly due to their involvement in data cleaning processes [19]. Among the different data cleaning activities, it is possible to find: schema mapping, de-duplication, classification and mastering, spotting errors and violations (e.g., outliers), repairing incorrect values, missing value imputation [16].

Data cleaning activities have also been included in several data preparation commercial tools [15]. Two of the main frameworks that can be used to perform data cleaning are: LLUNATIC [14] and HoloClean [22]. The first represents one of the first uniform frameworks for data-cleaning activities, which also introduced novel semantics and notions in the data cleaning research area. The latter relies on classical concepts, such as integrity constraints and external data sources, but also on statistical properties of the input data. The main goal of HoloClean is to automatically generate a probabilistic program that performs data cleaning.

Data cleaning and metadata verification activities have also been included into, mainly commercial, data governance suites, such as Talend [23] and IBM InfoShere [11]. Among the many advanced database operations, these suites permit to validate and possibly to graphically check data quality constraints.

The possibility of exploiting metadata is given by the existence of algorithms capable of discovering them from big data sources [1, 4, 10, 21]. Metadata discovery processes are also provided by effective platforms for data profiling, such as the Metanome project [20], which embeds several algorithms to automatically discover complex metadata, including functional and inclusion dependencies; and Metacrate [17], which permits the storage of different meta-data and their integration, enabling users to perform several ad-hoc analysis.

Nevertheless, since automatic discovery processes could output many metadata, some novel proposals started to visually manage the complexity related to an effective visualization of metadata holding on a given dataset, by using novel metaphors for representing metadata at different levels of detail [9]. Moreover, other proposals allow users to explore how metadata change over time, and to compare the results obtained among different time-slots [6], or to represent how discovery results change into the search space [7].

All of the above-mentioned approaches try to repair/analyze available datasets aiming at improving their quality and/or detect/visualize possible holding metadata. On the contrary, the proposed INDITIO plugin aims at improving the quality of data, trying to directly make users, i.e., who define data, aware of the possible errors that they can introduce into database instances. For this reason, we propose a tool that could be properly integrated into a DBMS. In this way, users and/or domain experts can approach this kind of concepts by simply using the environment they are familiar with. In other words, INDITIO aims at including data profiling discovery results within DBMSs, facilitating their interpretation to users through a plugin that also includes several visual components. To the best of our knowledge, in the literature, there are no other DBMS integrated plugins considering the interaction among data definition statements and metadata.

3 DATA PROFILING

Collecting metadata from big datasets is the goal of the data profiling research area [19]. Profiling metadata refer to many kinds of properties, ranging from statistics, domain cardinalities, frequent patterns to clusters, outliers, and data dependencies. All of them might be exploited in several advanced database operations, such as query optimization, data cleaning, and so forth. The proposed plugin considers two specific types of metadata, e.g., Unique Column Combination (UCC) and Functional Dependency (FD), which are described in what follows, after a brief introduction to relational databases.

A relational database schema $\mathcal{R}$ is defined as a collection of relation schemas $(R_1,\dots, R_n)$, where each R_i is defined over a set $attr(R_i)$ of attributes $(A_1,\dots, A_m)$. Each attribute A_k has associated a domain $dom(A_k)$, which can be finite or infinite. A relation instance (or simply a relation) r_i of R_i is a set of tuples $(t_1, \dots, t_p)$ such that $\forall\ A_k \in attr(R_i)\ t_j[A_k] \in dom(A_k)$, where $t_j[A_k]$ denotes the projection of t_j onto A_k. A database instance r of $\mathcal{R}$ is a collection of relations $(r_1,\dots,r_n)$, where r_i is a relation instance of R_i, for $i \in [1, n]$.

In the context of relational databases, one of the main property is represented by candidate keys. They permit to define possible tuple identifiers of a relation instance since no repetition in value combinations are allowed. They can be effectively identified by exploiting unique column combinations (UCCs).

UCC definition. A UCC over a relation schema R is a sets of attributes $K \subseteq attr(R)$ such that given an instance r of R, for every pair of tuples (t_1, t_2) in r then $t_1[K] \neq t_2[K]$.

Another relevant property of a relational database is represented by functional dependencies (FDs), which are used to improve the quality of database schemas and to reduce manipulation anomalies.

FD definition. An FD over a database schema $\mathcal{R}$ is a statement $X \rightarrow Y$ (X implies Y) defined between two sets of attributes $X, Y \subseteq attr(\mathcal{R})$, such that, given an instance r of $\mathcal{R}$, $X \rightarrow Y$ is satisfied in r if and only if for every pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$. X and Y represent the Left-Hand-Side (LHS) and Right-Hand-Side (RHS) of the FD, respectively.

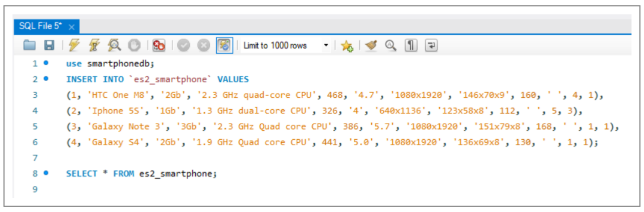
In general, an FD is said to be *non-trivial* if and only if $X \cap Y = \emptyset$. Moreover, an FD is said to be *minimal* if and only if there is no attribute $B \in X$ such that $X \backslash B \rightarrow Y$ holds on r.

4 INDITIO: A MYSQL WORKBENCH PLUGIN

In this section, we describe INDITIO a MySQL Workbench plugin capable of validating profiling metadata upon the insertion of new tuples in a database. MySQL Workbench[1] is the official graphical client of MySQL [3], which covers many functionalities, among which the most important are the possibility *i*) to graphically create models of database schemas, *ii*) to edit tables, columns, indices, and so forth, and *iii*) to create and manage connections to database servers, along with providing the capability to execute SQL queries using the built-in SQL Editor (see Figure 1).

INDITIO has been implemented by exploiting the libraries exposed from Oracle, to directly interact with the main components of the software. Thanks to the most recent versions of the MySQL Workbench source packages, it was possible to develop the proposed plugin using Python 2.7. Although this is not the latest version of this programming language, it is the only supported version by MySQL Workbench. In fact, it is important to notice that MySQL Workbench has been developed by exploiting C++ libraries, and supports Python libraries only by means of a wrapper that is able to translate Python code into C++ code. This makes

[1] https://www.mysql.com/it/products/workbench/



```
use smartphonedb;
INSERT INTO `es2_smartphone` VALUES
(1, 'HTC One M8', '2Gb', '2.3 GHz quad-core CPU', 468, '4.7', '1080x1920', '146x70x9', 160, ' ', 4, 1),
(2, 'Iphone 5S', '1Gb', '1.3 GHz dual-core CPU', 326, '4', '640x1136', '123x58x8', 112, ' ', 5, 3),
(3, 'Galaxy Note 3', '3Gb', '2.3 GHz Quad core CPU', 386, '5.7', '1080x1920', '151x79x8', 168, ' ', 1, 1),
(4, 'Galaxy S4', '2Gb', '1.9 GHz Quad core CPU', 441, '5.0', '1080x1920', '136x69x8', 130, ' ', 1, 1);

SELECT * FROM es2_smartphone;
```

Figure 1: The MySQL Workbench SQL Editor.

MySQL Workbench a real challenge for external developers looking to extend its functionalities, since it is not always possible to install new modules. For this reason, the most popular currently available plugins created by external developers [2] are divided into two categories: *i*) MySQL database documentation generation; and *ii*) MySQL Workbench export. Both types of plugins work in batch and are bound to execute a single operation. On the contrary, INDITIO fully exploits the functionalities of MySQL Workbench as described in the following section.

4.1 Interacting with metadata upon data insertions

INDITIO is able to intercept data insertion queries provided by users into the SQL Editor. Due to the possible big number of errors that can be introduced during the insertion of new tuples, DBMSs should enable users to visualize profiling metadata that newly inserted data could possibly invalidate, by also giving the possibility to interact with them. For this reason, it should be necessary to evaluate such metadata upon data insertion operations. To this end, INDITIO extends MySQL Workbench functionalities by enabling users to validate UCCs and FDs upon the insertion of new tuples.

Figure 2 shows the general visual interface of INDITIO. In general, it permits to evaluate the impact of new data on a set of holding metadata. Thus, it enables users to visualize the new tuples being inserted (Figure 2(a)), the metadata to validate (Figure 2(b-d)), and the results of the metadata validation process (Figure 2(e-h)). Moreover, INDITIO also provides several functionalities enabling users to interact with both SQL statements and metadata, as described in the following. A demonstration video of INDITIO is available on YouTube[3].

4.1.1 Analysis of metadata. The main novelty introduced by INDITIO is the possibility to evaluate some metadata directly into the MySQL Workbench. In fact, a user can upload UCCs or FDs by selecting the type of metadata s/he plans to monitor. In particular, the metadata will be directly uploaded through a default file, but they can be changed by selecting a new JSON file (Figure 2(b)).

The uploaded metadata are shown in the middle form (Figure 2(d)), which is customized according to the type of metadata the user selects. For instance, FDs are divided into LHS and RHS, each containing some attributes, in order to graphically visualize the implication property. Instead, the UCC customized form visualizes each metadata by considering a single group of attributes (Figure 3). Aside from the metadata uploaded via file, a user can always add new metadata. Moreover, through this form it is possible to select which metadata must be considered during the validation. Indeed, each metadata can be selected by means of the check box, and/or by using the "Select All" or "Unselect All" buttons. Finally, it is always possible to "Copy Selected Metadata" as text by means of a specific button.

Notice that metadata are described through letters or numbers, e.g., alias, in order to identify attributes. This facilitates users in focusing on attributes and/or in defining new metadata. In fact, possible long (or inappropriate) attribute names could confuse the user. However, INDITIO provides a suitable form to show the mapping between attribute names and their associated alias (see Figure 2(c)).

All selected metadata can be validated by clicking on "Run Validation", which triggers the execution of a validation module whose aim is to check if the new tuples violate the selected metadata. According to the validation process, each metadata can be classified in one of the following categories:

- *Valid Metadata*, when the new tuples do not produce any violation;
- *Not Valid Metadata*, when the new tuples entail at least one violation; or
- *Impossible to validate*, when the metadata cannot be validated. This occurs when the user introduces errors in the metadata, such as when the attribute names do not exist in the considered database.

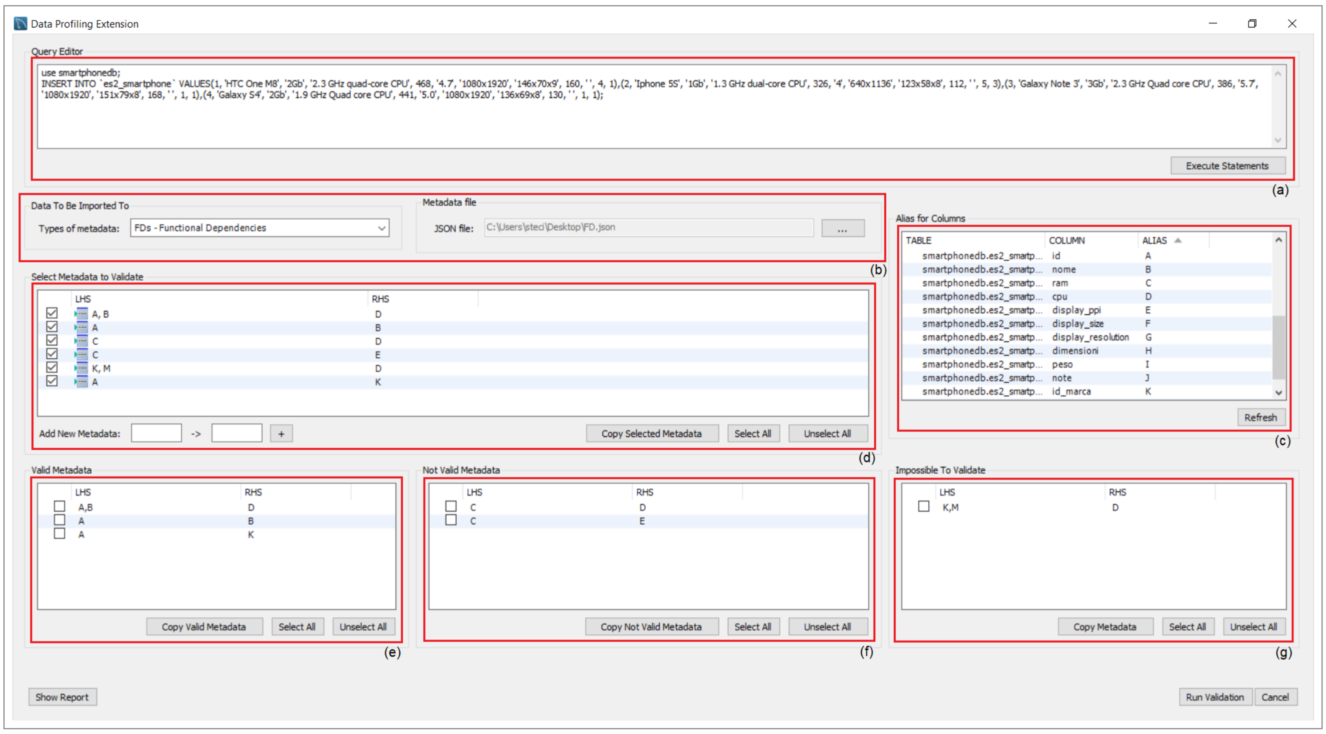
Example 4.1. Let us consider a database storing smartphone characteristics. Figure 2(e-g) show validation results of the considered FDs (Figure 2(d)) according to the new tuples the user is planning to insert (see Figure 2(a)). In particular, three out of six FDs are valid, two are invalid, and one cannot be validated. In fact, K,M ⟶ D includes attribute M that does not appear in the considered database. Instead, C ⟶ E, i.e., ram ⟶ display_ppi, is invalidated if the new tuples are inserted.

In general, the impact of the new tuples on the considered metadata is summarized by INDITIO in a new form, named *report form*, shown in Figure 4. This form graphically shows the percentage of validation/invalidation produced on the selected metadata by the tuples the user is planning to insert. Moreover, the report form ranks database attributes in descending order according to the number of invalidated metadata containing them. More specifically, the form represents this kind of information according to the type of metadata, i.e., by splitting the information about invalidation on LHS and RHS when considering FDs (see Figure 4(a)).

Example 4.2. Figure 4(a) shows the FD validation report for the validation results represented in Figure 2. In particular, the form shows that the impact of invalidations is 33% of the analyzed metadata. Moreover, among the attributes involved in the invalidated metadata, attribute C (e.g., ram) is involved in two invalidated FDs; whereas attributes E (e.g., cpu) and D (e.g., display_ppi) are involved in one invalidated FD. This could suggest to verify the values of attributes ram, cpu, and display_ppi of the new tuples.

4.1.2 Interacting with data insertions. INDITIO not only enables users to visualize the impact of new tuples on holding metadata, but it also permits them to interact with the new data. First of all, INSERT INTO statements can always be modified within the INDITIO interface (see Figure 2(a)), triggering subsequent validation processes with modified tuples. As said above, INDITIO freezes the execution of INSERT INTO statements while verifying the possibility to correct values being inserted so as not to invalidate holding metadata. Nevertheless, INDITIO always

[2]https://github.com/cciro94/MySQL-Workbench-Plugins
[3]https://youtu.be/u03Vftge8pA



Query Editor

```
use smartphonedb;
INSERT INTO `es2_smartphone` VALUES(1, 'HTC One M8', '2Gb', '2.3 GHz quad-core CPU', 468, '4.7', '1080x1920', '146x70x9', 160, '', 4, 1),(2, 'Iphone 5S', '1Gb', '1.3 GHz dual-core CPU', 326, '4', '640x1136', '123x58x8', 112, '', 5, 3),(3, 'Galaxy Note 3', '3Gb', '2.3 GHz Quad core CPU', 386, '5.7',
'1080x1920', '151x79x8', 168, '', 1, 1),(4, 'Galaxy S4', '2Gb', '1.9 GHz Quad core CPU', 441, '5.0', '1080x1920', '136x69x8', 130, '', 1, 1);
```

Execute Statements

Data To Be Imported To

Types of metadata: FDs - Functional Dependencies

Metadata file

JSON file: C:\Users\steci\Desktop\FD.json

Alias for Columns

TABLE	COLUMN	ALIAS
smartphonedb.es2_smartp...	id	A
smartphonedb.es2_smartp...	nome	B
smartphonedb.es2_smartp...	ram	C
smartphonedb.es2_smartp...	cpu	D
smartphonedb.es2_smartp...	display_ppi	E
smartphonedb.es2_smartp...	display_size	F
smartphonedb.es2_smartp...	display_resolution	G
smartphonedb.es2_smartp...	dimensioni	H
smartphonedb.es2_smartp...	peso	I
smartphonedb.es2_smartp...	note	J
smartphonedb.es2_smartp...	id_marca	K

Refresh

Select Metadata to Validate

	LHS	RHS
☑	A, B	D
☑	A	B
☑	C	D
☑	C	E
☑	K, M	D
☑	A	K

Add New Metadata: ->

Copy Selected Metadata · Select All · Unselect All

Valid Metadata

	LHS	RHS
☐	A,B	D
☐	A	B
☐	A	K

Copy Valid Metadata · Select All · Unselect All

Not Valid Metadata

	LHS	RHS
☐	C	D
☐	C	E

Copy Not Valid Metadata · Select All · Unselect All

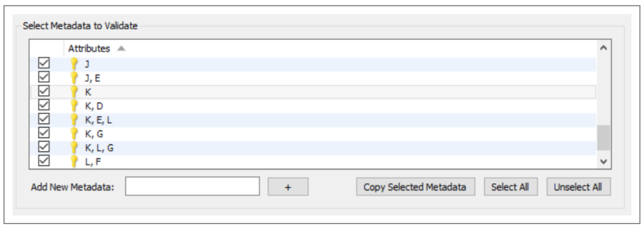
Impossible To Validate

	LHS	RHS
☐	K,M	D

Copy Metadata · Select All · Unselect All

Show Report · Run Validation · Cancel

Figure 2: The INDITIO visual interface.



Select Metadata to Validate

	Attributes
☑	J
☑	J, E
☑	K
☑	K, D
☑	K, E, L
☑	K, G
☑	K, L, G
☑	L, F

Add New Metadata:

+ · Copy Selected Metadata · Select All · Unselect All

Figure 3: Visualization of UCCS.

gives users the possibility to overlook possible violations of metadata and to force the execution of data insertion operations by means of the "Execute Statements" button (see Figure 2(a)).

On the other hand, one of the main goals of INDITIO is to help users in correcting possible errors. To this end, after a validation process (i.e., by clicking "Run Validation"), a user can visualize data yielding violations by interacting with the "Not Valid Metadata" form (see Figure 2(f)). More specifically, by clicking on any metadata in such form, INDITIO shows a *violation detail* form, as shown in Figure 5. In particular, concerning FDS, apart from the details of the selected metadata, the violation form describes for each LHS value combination involved in a violation: *i*) the value combination of the LHS, *ii*) the corresponding distinct values found on the RHS, and *iii*) their total number (see Figure 5(a)). Instead, concerning UCCS, apart from the details of the selected metadata, the violation form describes for each value combination involved in value duplication: *i*) the value combination involved in a duplication, *ii*) the number of duplications (see Figure 5(b)). The latter should represent the functionality that drives users in accomplishing the best possible correction of errors.

Example 4.3. Figure 5(a) shows the violation details of the FD C → E (e.g. ram → display_ppi) according to the validation results represented in Figure 2. In particular, the form shows that four specific values on attribute C (e.g. ram), i.e. 1Gb, 2Gb, 3Gb, 512mb, each implies different values of attribute E (e.g. display_ppi). Moreover, it is also possible to see that the value 2Gb is the one implying the highest number of distinct values. Instead, Figure 5(a) shows that for the UCC K,E,L (e.g. id_brand, display_ppi, id_os) there are six specific value combinations inducing duplicate values. In general, this form could suggest to correct values on the new tuples concerning attributes involved in the considered violated metadata, or to exclude the metadata from the validation process.

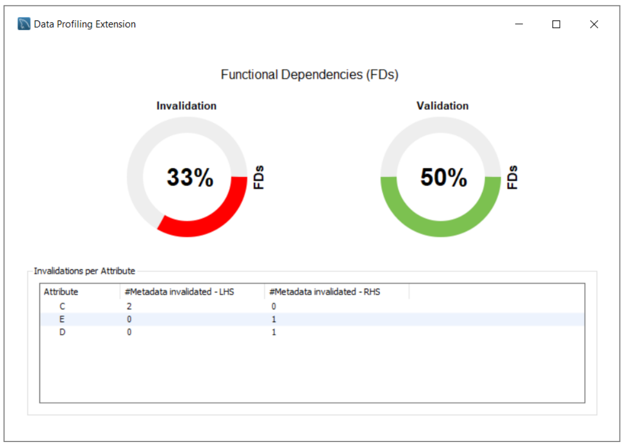
5 USER STUDY

The user study presented in this section aims to show that INDITIO makes metadata validation a simple and effective process for improving data quality.

5.1 Method

We recruited 86 students majoring in Computer Science who just attended the *Fundamentals of database*. We also recruited 3 Ph.D. students and 1 Ph.D., all of which were familiar with the given domain. Statistics about participants have been collected through a background survey, as shown in Table 1(Q1-Q4), and whose results are reported in Figure 6. In particular, about 85% of the recruited people were men, 15% were women, and most of them were undergraduate students. Moreover, on average they declared, through a Likert scale, a medium level of knowledge concerning MySQL and MySQL Workbench. Before the evaluation started, participants underwent a 45-min tutorial on the theoretical foundations of data profiling and INDITIO.

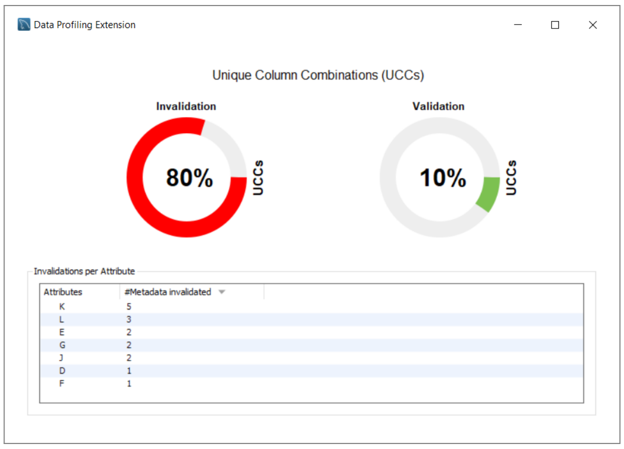
Each of the 90 participants was given a database concerning personal data, three data insertion statements, and two sets of UCCS and FDS metadata, respectively. Moreover, we requested



Invalidations per Attribute

Attribute	#Metadata invalidated - LHS	#Metadata invalidated - RHS
C	2	0
E	0	1
D	0	1

(a) A form showing validation statistics after the FD validation process.

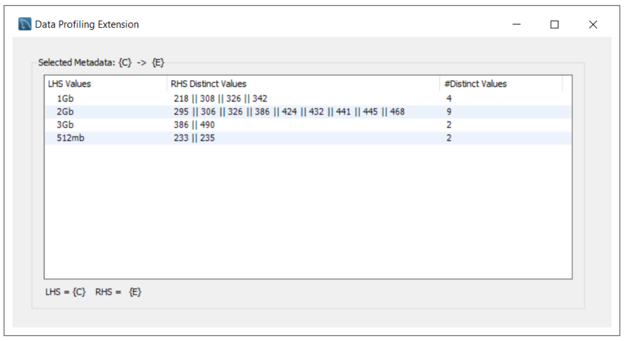


Invalidations per Attribute

Attributes	#Metadata invalidated
K	5
L	3
E	2
G	2
J	2
D	1
F	1

(b) A form showing validation statistics after the UCC validation process.

Figure 4: Validation statistics provided by INDITIO.

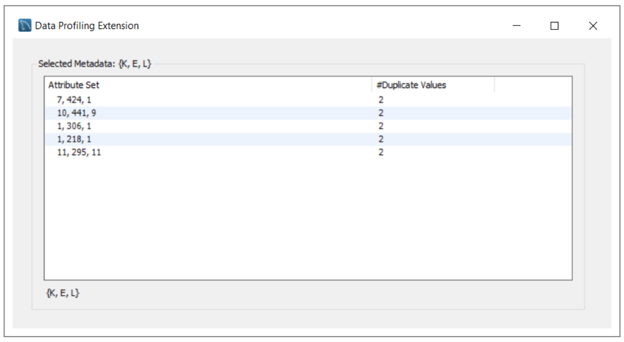


Selected Metadata: {C} -> {E}

LHS Values	RHS Distinct Values	#Distinct Values
1Gb	218 || 308 || 326 || 342	4
2Gb	295 || 306 || 326 || 386 || 424 || 432 || 441 || 445 || 468	9
3Gb	386 || 490	2
512mb	233 || 235	2

LHS = {C} RHS = {E}

(a) A form showing values that invalidate metadata after the FD validation process.



Selected Metadata: {K, E, L}

Attribute Set	#Duplicate Values
7, 424, 1	2
10, 441, 9	2
1, 306, 1	2
1, 218, 1	2
11, 295, 11	2

{K, E, L}

(b) A form showing values that invalidate metadata after the UCC validation process.

Figure 5: Violation details of INDITIO.

them to check the correctness of data insertion statements according to the provided metadata, and if necessary, to correct statements aiming to guarantee the validity of the provided metadata. More specifically, we conducted a within subjects study by considering two scenarios: with and without INDITIO, and requested to accomplish the task in one scenario first, and then with the other one. Half participants considered first the scenario without INDITIO, while the remaining ones used INDITIO first. Notice that, the provided data insertion statements were different but equivalent in complexity. In particular, in the case participants performed the tasks without the tool, they were able to validate the metadata by directly analyzing the data source or using the SQL language to compose specific queries. To this end, they could interact only with the tools already integrated into MySQL Workbench, both to validate the metadata and to correct the values within the data.

After completing the assigned tasks, participants were requested to fill some questionnaires, aiming to highlight the advantages and drawbacks of INDITIO (see Table 1(Q4-Q20)). More specifically, questions from Q4 to Q8 have been filled after participants performed each task (with and without INDITIO, or vice versa), whereas the remaining ones have been included in a final survey. Moreover, questions from Q4 to Q20 are quantitative, and they have been measured through a Likert scale, ranging from 1, mapping "Strongly disagree" response, to 5, mapping "Strongly agree" response. Finally, to further evaluate the effectiveness of both processes (with and without INDITIO) we measured the time required for completing the task and the number of errors.

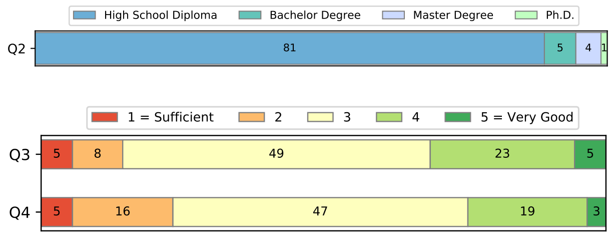


Figure 6: Statistics concerning involved participants.

5.2 Results and Discussion

Figure 8 shows the results achieved from each participant while executing the assigned tasks with both the compared scenarios (without and with the tool). The results achieved from the tasks performed without the tool have been considered as the comparative baseline (purple line), while the results obtained with the

Survey	Alias	Question
Background	**Q1**	Gender
	Q2	Qualification
	Q3	Level of knowledge of MySQL
	Q4	Level of knowledge of MySQL Workbench
Comparative	**Q5**	I completed the tasks quickly and easily
	Q6	The instructions for completing the tasks are clear and easy to read
	Q7	The metadata validation process has been simple
	Q8	The values that invalidated the metadata have been easy to find
Final	**Q9**	The tool is simple to use
	Q10	The tool is simple to learn
	Q11	I was able to retrieve back the process, whenever I made a mistake
	Q12	The tool shows the information very clearly
	Q13	The tool is pleasant to use
	Q14	The user interface is pleasant and informative
	Q15	The tool made transparent the execution of the underlying validation processes
	Q16	It was simple to understand the metadata syntax
	Q17	The tool presents all the features I expected
	Q18	I am in general satisfied about the tool
	Q19	In the future, I would like to use the tool
	Q20	The tool simplified the validation process with respect to the manual process
	Q21	What is your general impression about the tool?
	Q22	Do you have any improvements to suggest?
	Q23	Which feature of the tool did you like the least?
	Q24	Which feature of the tool did you like the most?

Table 1: Questions proposed to participants.

proposed tool were described by the bars. In particular, the plot highlights the improvement obtained by using INDITIO, e.g., a value of 2 indicates that the results achieved with INDITIO are 2 times better than those achieved without it, whereas a value less than 1 indicates the opposite case. In general, it can be observed that most of the participants performed better with INDITIO, even if satisfactory results have been achieved also without the use of the plugin. Concerning the time employed to complete the assigned tasks, on average participants took thirty minutes with INDITIO, ranging from five to sixty-seven minutes, and forty-five minutes without it, ranging from five to eighty-four minutes. In general, we noticed that the manual validation task times were particularly long, especially for users with less knowledge of MySQL. On the contrary, with INDITIO almost all users have reduced the time of the validation processes by more than 50%.

Figure 7 depicts the box plots derived from the answers (on a Likert scale from 1 to 5) to questions ranging from Q5 to Q8. In particular, users answered the same questions after performing tasks for each considered scenario (with and without the tool). A boxplot shows the median (horizontal lines), the interquartile ranges (boxes), the largest and the smallest observations (whiskers). By comparing the results achieved without the plugin (the first four box plots in Figure 7) against those with the plugin, (the remaining box plots) we can conclude that participants felt more comfortable and effective when working with INDITIO.

Concerning the quantitative questions in the final questionnaire, INDITIO obtained the general agreement of participants, while evaluating its usability and effectiveness (see Figure 9). In particular, according to answers for Q20, the capability of simplifying the metadata validation process has been widely recognized to INDITIO. The latter has turned out to be comfortable and useful according to answers to questions to Q18 and Q19, simple to learn, and pleasant to use (see Figure 9: answers for Q10 and Q13). Nevertheless, some work should be made to further improve the general usability of the plugin and the transparency of the validation process, according to answers to questions to Q11, Q12, and Q15.

The open questions in the final questionnaire (see Q21-Q24 in Table 1) aimed at highlighting the strengths and weakness of INDITIO. In particular, concerning the general impressions about the plugin (see Q21 in Table 1), many participants said that INDITIO shows a simple and intuitive interface, in which the components appear well organized in the frame. Another part of them expressed their opinion on the usefulness and efficiency of the

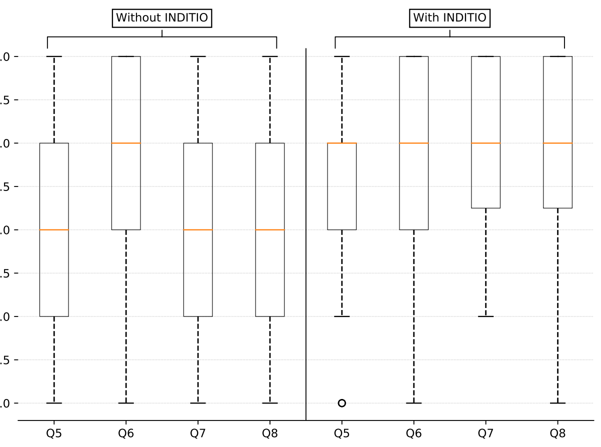


Figure 7: Comparative boxplots showing distribution of user answers to the quantitative questionnaire.

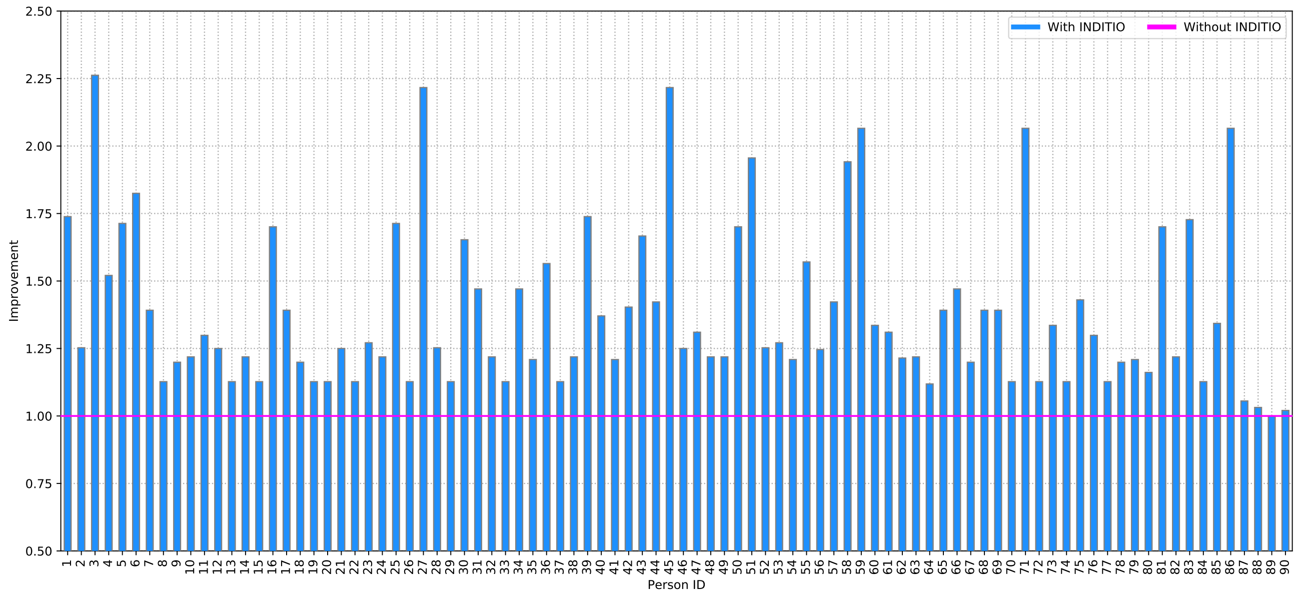


Figure 8: Distributing scores achieved by participants for each analyzed scenario (with and without INDITIO).

plugin, claiming that "*The tool has certainly been successful in its intent, significantly speeding up the time for validating metadata*". Moreover, some of the most interesting comments have been provided from users that are less familiar with the research context and with MySQL Workbench. In fact, they report that INDITIO is able to improve the understanding of MySQL Workbench and to show that metadata potentially allows them to extract further knowledge from data, which often is not clearly visible. Only a small part of inexperienced users asserted that the plugin interface might be initially unclear, claiming that "*Initially, the tool seems difficult to understand. Then, once I understand how to use it, it is very useful for checking the validity of FD and UCC*". However, by practising with it, INDITIO allowed them to easily understand its features and to become familiar with its environment.

In addition, concerning the specific features that participants liked the most and the least (see Q22-Q23 in Table 1), experienced users have greatly appreciated the functionality of identifying values yielding the invalidation of metadata. In fact, some of them claimed that "*One of the most interesting features is identifying the values that invalidate the metadata. This functionality could be directly integrated into the MySql Workbench suite.*" Instead, inexperienced users have shown interest for the report forms (see Figure 5) and for the simplicity through which the plugin could be integrated within the MySQL Workbench suite as a simple plugin. Although most of the comments were positive, we also investigated the features they liked the least. Among them, many users have highlighted that the Query Editor component appears small and does not clearly show statements. However, this is limited by the graphics components included in the MySQL Workbench. For these reasons, we allow the users to directly interact with the SQL editor of MySQL Workbench, and to import their statements. Other participants have suggested adding further reports in the interface in order to enhance their understanding of how data insertion statements affect metadata. Only few users have proposed to extend the interface of INDITIO with new graphical components in order to improve the interaction with both the plugin and the MySQL Workbench.

Finally, we have asked users some suggestions for enhancing INDITIO (see Q24 in Table 1). To this end, some users have suggested integrating new metadata, also allowing them to simultaneously validate multiple metadata. Other users have suggested improving the integration with systems based on the Linux architecture. In fact, it has been found that some of the users using these operating systems tend to view some reports differently from users who use Windows systems. However, this is due to the compatibility problems between the technologies underlying the MySQL Workbench and different operating systems. In the future, these compatibility issues might be solved with new software versions.

In summary, the four open questions of the final questionnaire revealed that some participants remarked some limitations of the INDITIO user interface. Moreover, they would like to receive more hints during the statement modification process, according to validation results. Conversely, they positively judged the intuitiveness the metadata validation and the error detection processes. Moreover, they welcomed the tool and recognized its usefulness.

6 CONCLUSION

In this paper we presented INDITIO, a MySQL plugin enabling users to assess the quality of data they planned to insert into a database. In particular, the plugin enables a user to observe how the new tuples can affect the validation of some metadata, e.g. FDS and UCCS, assisting him/her in correcting them, if needed. We evaluated INDITIO by involving more than eighty participants in a user study, which demonstrated its usefulness, giving us the possibility to detect the main characteristics of the tool to be improved, mainly focusing on the improvement and/or extension of some visual components.

The potentiality of INDITIO yields many possible future directions. In particular, we are currently working on the enrichment of the set of possible metadata that can be taken into consideration, with particularly emphasis on relaxed FDS (RFDS) [8].

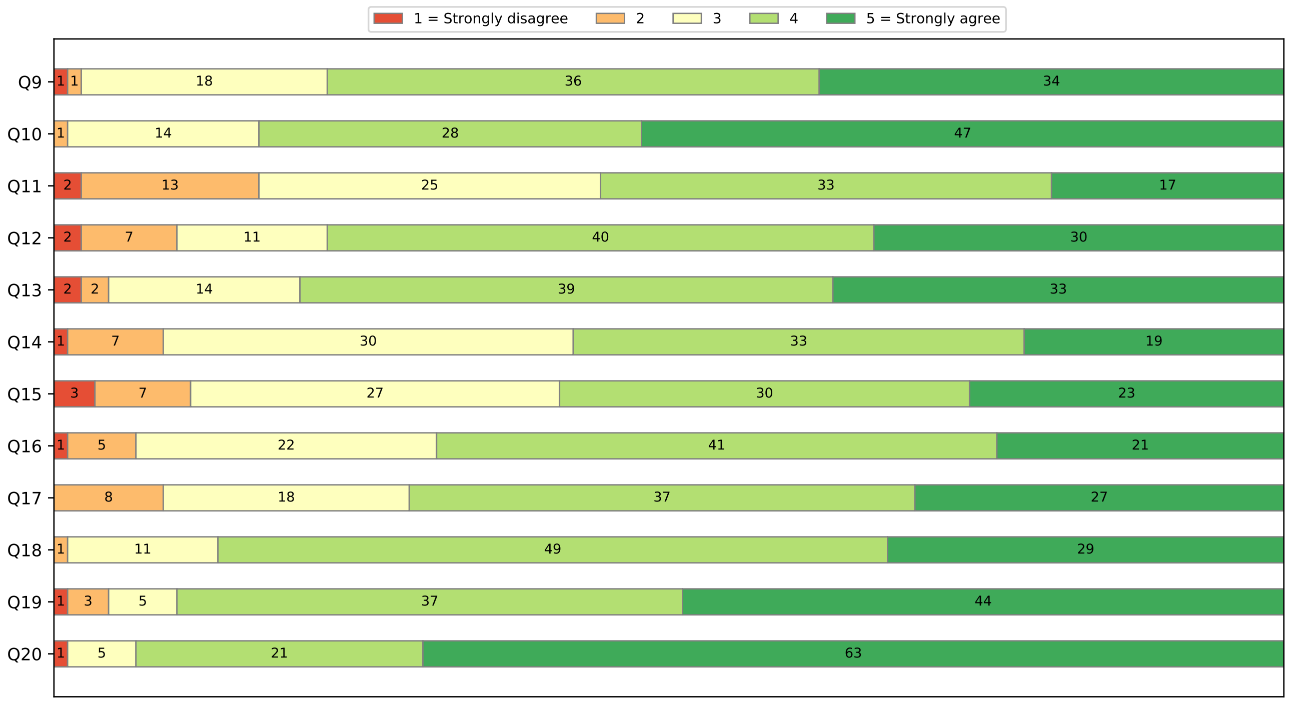


Figure 9: Distributing participant answers to the quantitative questions in the final questionnaire.

Moreover, we would like to directly connect the plugin to incremental or continuous discovery algorithms [7], so that when the user decides to force the insertions also in presence of invalidated metadata, such algorithms could automatically update the set of holding metadata. This would implicitly represent a means to automatically analyze the significance of metadata, while discovering them from actual data.

REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. 2018. Data profiling. *Synthesis Lectures on Data Management* 10, 4 (2018), 1–154.

[2] Ibrahim Ahmed Al-Baltah, Abdul Azim Abd Ghani, Ghilan Mohammed Al-Gomaei, Fua'ad Hassan Abdulrazzak, and Abdulmonem Ali Al Kharusi. 2020. A scalable semantic data fusion framework for heterogeneous sensors data. *Journal of Ambient Intelligence and Humanized Computing* (2020), 1–20.

[3] Oracle Corporation and/or its affiliates. 2021. MySQL™ Workbench Reference Manual. https://dev.mysql.com/doc/workbench/en/. last accessed: Jan 13th, 2021.

[4] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment* 11, 3 (2017), 311–323.

[5] Jens Bleiholder and Felix Naumann. 2009. Data fusion. *ACM computing surveys (CSUR)* 41, 1 (2009), 1–41.

[6] Bernardo Breve, Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. 2020. Visualizing Dependencies during Incremental Discovery Processes. In *Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference (CEUR Workshop Proceedings)*, Vol. 2578. CEUR-WS.org, Aachen, 1–8.

[7] Loredana Caruccio and Stefano Cirillo. 2020. Incremental discovery of imprecise functional dependencies. *Journal of Data and Information Quality (JDIQ)* 12, 4 (2020), 1–25.

[8] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2016. Relaxed Functional Dependencies – A Survey of Approaches. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 147–165.

[9] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2019. Visualization of (multimedia) dependencies from big data. *Multimedia Tools and Applications* 78, 23 (2019), 33151–33167.

[10] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2020. Mining relaxed functional dependencies from data. *Data Mining and Knowledge Discovery* 34, 2 (2020), 443–477.

[11] IBM Corporation. 2021. IBM InfoSphere Information Server. https://www.ibm.com/analytics/information-server/. last accessed: Feb 23th, 2021.

[12] Wei Dai, Isaac Wardlaw, Yu Cui, Kashif Mehdi, Yanyan Li, and Jun Long. 2016. Data profiling technology of data governance regarding big data: review and rethinking. In *Information Technology: New Generations*. Springer International Publishing, Cham, 439–450.

[13] Wenfei Fan, Floris Geerts, and Xibei Jia. 2008. Semandaq: a data quality system based on conditional functional dependencies. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1460–1463.

[14] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC data-cleaning framework. *Proceedings of the VLDB Endowment* 6, 9 (2013), 625–636.

[15] Mazhar Hameed and Felix Naumann. 2020. Data Preparation: A Survey of Commercial Tools. *SIGMOD Rec.* 49, 3 (2020), 18–29.

[16] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM, New York, NY, United States.

[17] Sebastian Kruse, David Hahn, Marius Walter, and Felix Naumann. 2017. Metacrate: Organize and analyze millions of data profiles. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 2483–2486.

[18] Jay Lee, Hung-An Kao, Shanhu Yang, et al. 2014. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp* 16, 1 (2014), 3–8.

[19] Felix Naumann. 2014. Data profiling revisited. *ACM SIGMOD Record* 42, 4 (2014), 40–49.

[20] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. 2015. Data profiling with Metanome. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1860–1863.

[21] Thorsten Papenbrock and Felix Naumann. 2016. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 821–833.

[22] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017), 1190–1201.

[23] Talend. 2021. Talend Studio. https://https://www.talend.com/products/data-quality/. last accessed: Feb 23th, 2021.