QuaIIe: A Data Quality Assessment Tool for Integrated Information Systems

Lisa Ehrlinger*†
†Software Competence Center Hagenberg
Softwarepark 21, 4232 Hagenberg, Austria
email: lisa.ehrlinger@scch.at

Bernhard Werth*, Wolfram Wöß*
*Johannes Kepler University Linz
Altenberger Straße 69, 4040 Linz, Austria
email: {lisa.ehrlinger, wolfram.woess}@jku.at

***Abstract*—Data is central to decision-making in enterprises and organizations (e.g., smart factories and predictive maintenance), as well as in private life (e.g., booking platforms). Especially in artificial intelligence applications, like self-driving cars, trust in data-driven decisions depends directly on the quality of the underlying data. Therefore, it is essential to know the quality of the data in order to assess the trustworthiness and to reduce the uncertainty of the derived decisions. In this paper, we present QuaIIe (Quality Assessment for Integrated Information Environments, pronounced ['kvɑlə]), a Java-based tool for the domain-independent ad-hoc measurement of an information system's quality. QuaIIe is based on a holistic approach to measure both schema and data quality and covers the dimensions accuracy, correctness, completeness, pertinence, minimality, and normalization. The quality measurements are presented as machine- and human-readable reports, which can be generated periodically in order to observe how data quality evolves. In contrast to most existing data quality tools, QuaIIe does not necessarily require domain knowledge and thus offers an initial ad-hoc estimation of an information system's quality.**

***Index Terms*—Data Quality; Information Integration; Estimation; Measurement; Trust.**

I. Introduction

Decision-making is usually based on data. Applications are process data in industry, sales, weather forecast, search engines, self-driving cars, or booking platforms. In order to trust such data-driven decisions, it is necessary to know the quality of the underlying data. Despite the clear correlation between data and decision quality, 84 % of the CEOs in the US are concerned about their data quality [1]. In addition to incorrect decision making, poor Data Quality (DQ) may cause effects like cost increase, customer dissatisfaction, and organizational mistrust [2]. According to an estimation by IBM, the total financial impact of poor quality data on business in the US was $3.1 trillion [3] in 2016. Thus, DQ is no longer a question of "hygiene", but has become critical for operational excellence and is perceived as the greatest challenge in corporate data management [4].

In practice, data of enterprises and organizations are often stored in Integrated Information Systems (IISs), which gather data from different and often heterogeneous information sources [5]. If such a system is queried, it is desirable to select the most appropriate and most trustworthy source with respect to query. Thus, an automated on-the-fly estimation of the eligible Information Sources (ISs) is necessary to judge weather an IS is complete or accurate enough to answer the query sufficiently. For this purpose, we developed QuaIIe (Quality Assessment for Integrated Information Environments), a modular Java-based tool that automatically performs quality measurement at the data-level and the schema-level. QuaIIe offers metrics for the quality dimensions accuracy, correctness, completeness, pertinence, minimality, and normalization.

Although the most frequently used definition of data quality is "fitness for use" [6], which expresses the high subjectivity and context-dependency of this concept, we aim at a domain-independent measurement of the quality of ISs. QuaIIe performs an automated ad-hoc estimation of the qualitative condition of multiple information sources within an IIS and generates a machine- and human-readable XML (extensible markup language) quality report. Such a report can be generated periodically in order to observe how DQ evolves. Our focus is the quality measurement of an IIS in productive use, and automatic data cleansing activities are therefore outside the scope of this research work. In a first step, it is essential to know the quality of the data in order to define goals and to verify the effectiveness of data cleansing activities.

The main contribution of this paper is the presentation of a novel tool that implements automated DQ measurement and estimation, covering the most important dimensions for both, data- and schema-level. To the best of our knowledge, there exists no tool that offers DQ metrics for such a large number of different DQ dimensions in a single application and comprises both data and schema quality. Therefore, we developed QuaIIe to fill this gap. The advantage of the presented approach is a long term observation of the DQ development, which provides indicators for further DQ improvements and thus, increases trustworthiness for data-driven decisions.

This paper is organized as follows: in Section 2 we discuss existing DQ tools and highlight their differences to QuaIIe. Section 3 covers the data and schema quality measurement, which was applied in this research. The implementation of QuaIIe is described, demonstrated and discussed in Section 4.

II. Related Work

Although the interest into DQ, from both research and industry, has increased over the last decade, it is still an underestimated topic in operational information systems. This fact is also reflected by the current market of DQ tools, which is considered a niche market despite its continuous growth [7]. In the following paragraphs, we give a short overview on existing DQ tools and discuss their differences to QuaIIe.

Gartner lists 39 commercial DQ tools by 16 vendors in their "Magic Quadrant of Data Quality Tools 2017" [7]. Most of the tools offer functionalities to investigate the qualitative condition of different data sources, manage DQ rules, resolve DQ issues, enrich data quality by integrate external data, validate addresses, standardize and cleanse data, and link related data entries using a variety of techniques. The aim of these commercial tools is usually the support of a comprehensive DQ program that involves management, IT, and business users. Thus, the application of such a tool usually requires a domain expert and preparatory work to be effective.

In addition to commercial DQ tools, a number of scientific tools has been proposed over the years, where the most important ones are compared and discussed in [8][9]. Both surveys make clear that the focus of those tools is on the detection and cleansing of specific DQ problems (e.g., name conflicts, missing data). QuaIIe, in contrast, focuses on the pure measurement (detection) of DQ problems and does not cleanse data, but with the advantage to be unsupervised, domain-independent and applicable for ad-hoc analysis. Additionally, and in contrast to most existing DQ tools, QuaIIe addresses the DQ topic from the dimension-oriented view. While a lot of research on DQ dimensions and their definition has been proposed in literature [2][6][10], there is no tool that implements metrics for such a broad number of dimensions. QuaIIe fills this gap and can thus be considered a vital complement in the section of research-oriented DQ tools. The main contributions in QuaIIe are (1) the combination of data and schema quality measurement and (2) the implementation of such a wide spectrum of different quality dimensions. Of course, more specialized tools might outperform QuaIIe in specific implementations, like distance calculation or string matching.

III. Data and Schema Quality Measurement

Data quality is usually described as multidimensional concept, which is characterized by different aspects, so called *dimensions* [6]. Those dimensions can either refer to the data values (i.e., *extension* of the data), or to their schema (i.e., the *intension* or data structure) [11]. While the majority of research into DQ focuses on the data values, QuaIIe covers dimensions for both schema and data quality. In fact, schema quality has a strong impact on the quality of the data values [11]. An example are redundant schema elements, which can lead to data inconsistencies. Thus, it is essential to consider both topics in order to provide holistic DQ measurement.

Since a wide variety of quality dimensions has been proposed over the years, we focus in the following paragraphs on accuracy, correctness, completeness, pertinence, minimality, and normalization. Each dimension can be quantified using one or several metrics, which capture the fulfillment of a dimension in a numerical value [12]. Some metrics require a reference or benchmark (*gold standard*) for their calculation. According to the Oxford Dictionary, a Gold Standard (GS) is "the best, most reliable, or most prestigious thing of its type" [13]. In the vast majority of cases a gold standard does not exist, but if there is one, it would be used in place of the IS under investigation. Thus, in practice, an existing benchmark is employed as gold standard, e.g., a single IS can be compared to the integrated data from the complete IIS. Although in practice, there is usually no complete gold standard for large data sets available, there are often reference data sets of good quality for a subset of the data. Examples are purchased reference data sets for customer addresses or a manually cleaned part of the original data. The quality estimation in QuaIIe (cf. Section III-F) allows to extrapolate the exact measurement for a part of the data to other parts that are required for a query but have not been yet measured. For more details to the schema quality dimensions applied in this paper, we refer to [14] and more information on the DQ dimensions can be found in [15].

A. Accuracy and Correctness

The terms *accuracy* and *correctness* are often used synonymously in literature and a number of different definitions exist for both terms [6][11][16]. In the DQ literature, accuracy can be described as the closeness between an information system and the part of the real-world it is supposed to model [11]. From the natural sciences perspective, accuracy is usually defined as the magnitude of an error [16]. In this research work, we refer to correctness for a calculation, which has been presented by Logan et al. [17], who distinguish between correct (C), incorrect (I), extra (E) and missing (M) elements after comparing a data set to its reference:

$$Cor(c, c') = \frac{C}{C + I + E}. \tag{1}$$

Here, the data correctness of, for instance, a relational table or class in an ontology, denoted as concept c, is measured by comparing it to its "correct" version c'. In this notion, C is the number of elements that correspond exactly to an element from the reference c'. The incorrect elements I have a similar element in the gold standard, but are not identical. While M describes the number of missing elements in the IS under investigation that exist in the gold standard, its complement E is the number of extra elements that exist in the investigated IS, but have no corresponding element in the gold standard. We refer to the values as CIEM counts.

In QuaIIe, however, an accuracy metric is implemented, which has its origins in the field of machine learning and is usually used to measure the accuracy of classification algorithms [18]. This accuracy metric can also be mapped to the notion by Logan et al. [17]:

$$Acc(c, c') = \frac{|c|}{|c \cup c'|} = \frac{C}{C + I + E + M} \tag{2}$$

where $|c|$ gives the number of records in a data set or concept c. In the rest of this paper, we refer to accuracy when discussing quality metrics for data values (since QuaIIe implements the metric for accuracy on data-level), and to correctness when discussing the corresponding schema dimension.

On the schema-level, Vossen [19] describes a database (DB) schema as correct, if the concepts of the related data model are applied in a syntactically and semantically appropriate way. Thus, he considers the model (e.g., Entity-Relationship model) as reference, which is assumed to be correctly available. In [11], the authors distinguish between correctness with respect to the model and with respect to the requirements. The correct representation of the schema requirements are considered a manual task, because requirements are rarely available in machine-readable form. Despite unknown quality, the content of an IS can be added as third possibility to validate a schema, in order to measure whether a schema fits its values. This includes for instance the correct usage of attributes (e.g., an attribute `first_name` actually contains a person's first name and no numeric value).

In QualIe, the formula by Logan et al. [17] for data correctness is also employed as a metric for schema correctness with C_s, I_s, E_s, and M_s denoting the correct, incorrect, extra, and missing elements of a schema s:

$$Cor(s, s') = \frac{C_s}{C_s + I_s + E_s}. \tag{3}$$

B. Completeness

Completeness is broadly defined as the breadth, depth, and scope of information contained in the data [10]. A number of authors [6][11] calculate data completeness according to:

$$Com(c, c') = \frac{|c|}{|c'|}. \tag{4}$$

Despite differences in expressions, most existing completeness metrics are correspondent to (4) and compare the number of elements in a data set $|c|$ to the number of elements in the gold standard $|c'|$. In this metric, scope for interpretation lies in selecting the gold standard or reference c' and in the similarity calculation (i.e., determining whether an element has a reference element in c'). In QualIe however, extra records, which exist in the gold standard, but have no counterpart in the data set under investigation are excluded and therefore have no influence on the completeness calculation. We use the formula presented by Logan et al. [17]:

$$Com(c, c') = \frac{C + I}{C + I + M}. \tag{5}$$

Schema completeness describes the extent to which real-world concepts of the application domain and their attributes and relationships are represented in the schema [11]. The metric for schema completeness in QualIe corresponds to the metric for data completeness in (5):

$$Com(s, s') = \frac{C_s + I_s}{C_s + I_s + M_s}. \tag{6}$$

Batista and Salgado [20] applied a schema completeness metric, which is equivalent to the data completeness in (4). In the calculation, the number of elements in the reference schema $|s'|$ is determined by counting the number of distinct elements in all schemas of an IIS. While the authors in [20] assume pre-defined schema mappings to be provided, QualIe implements the distance or similarity calculation between the schema elements on-the-fly.

In addition, Nauman et al. [21] proposed a comprehensive IIS completeness metric, which incorporates the *coverage* (i.e., data completeness of the extension of an IS), and *density* (i.e., schema completeness of the intension of an IS). The authors use the entire IIS as gold standard. The density of a schema is calculated according to the population of attributes with non-null values [21]. In contrast, the schema completeness metric in QualIe implements a data-value-independent calculation, which considers the existence of specific schema elements (e.g., relations in a relational DB).

C. Pertinence

Pertinence on the data-level equates to the notion of precision (in contrast to recall [18]) from the information retrieval field and complements data completeness. Data pertinence describes the prevalence of unnecessary records in the data. The classic precision metric is defined as the probability to select a correct element from a list [18] and in terms of correct, incorrect, extra, and missing records, is defined as:

$$Per(c, c') = \frac{C + I}{C + I + E}. \tag{7}$$

Schema pertinence describes a schema's relevance, which means that a schema with low pertinence has a high number of unnecessary elements [11]. A schema that is perfectly complete and pertinent represents exactly the reference schema (i.e., its real world representation), which means that the two dimensions complement each other. In accordance to (7), schema pertinence is calculated in QualIe as

$$Per(s, s') = \frac{C_s + I_s}{C_s + I_s + E_s}, \tag{8}$$

where the number of schema elements with a (correct or incorrect) correspondence in the gold standard is divided by the total number of elements in the schema under investigation.

D. Minimality

Information sources are considered minimal if no parts of them can be omitted without losing information, that is, the IS is without redundancies and no duplicate records exist [11]. The detection of duplicate records is a widely researched field that is also referred to as record linkage, data deduplication, data merging, or redundancy detection [22]. In order to determine which records of a data set are duplicates, different approaches exist. The most prominent approaches can be assigned to one of the following types [22]: (1) *probabilistic assignment* using the Fellegi-Sunter model [23], (2) *machine learning techniques* like support vector machines, clustering algorithms, or decision trees, (3) *distance-based methods*, which are based on a function that calculates the distance between two objects, and (4) *rule-based methods*, which are usually based on the work of domain experts.

In QualIe, duplicate detection is done by hierarchical clustering, which requires a distance function between the records. A distance function $\delta : o \times o \rightarrow [0, 1]$ is a function from a pair of elements to a normalized real number expressing the distance or dissimilarity between the two elements [24]. Analogous, some techniques calculate the similarity $\sigma : o \times o \rightarrow [0, 1]$ between two elements, which can be transformed to a distance value using the formula $\delta = 1 - \sigma$.

Since each data record consists of multiple attribute values, the distance function is a weighted-average of individual attribute distance functions. QualIe offers the following distance functions for data values: `AffineGapDistance`, `CosineDistance`, `LevenshteinDistance`, and `SubstringDistance` for strings, `AbsoluteValueDistance` for double values, `EqualRecordDistance` for entire records, as well as `EnsembleDistance` for any data type. The latter one combines an arbitrary number of other distances and a weight for each one. Thus, it allows the creation of distances that are adjusted to a specific IS schema, for example, to calculate the distance between persons by applying a string distance to the first and last name and a distance for numeric attributes to the age, and giving higher weights to the name than the age.

The main advantage of clustering in our approach is the automatic resolution of multiple correspondences. It thus, however, requires a threshold to be defined. QualIe sets a predefined clustering threshold which has been evaluated in experiments presented in [14]. In an automated test run, similarity matrices with different parameter combinations have been compared to a similarity matrix created by a domain expert using the mean squared error (MSE). The parameter combination yielding the closest similarity results (having a MSE of 0.0102) were used as standard parameters. However, QualIe also allows to overwrite those values by the user to adjust for specific domains. Hierarchical clustering initially creates one cluster for each observed record and continuously combines different clusters until all records are subsumed into one large cluster. QualIe offers seven different linkage strategies (single linkage, complete linkage, median linkage, mean linkage, pair group method with arithmetic mean, centroid linkage, and Ward's method). We refer to [25] for in-depth information on hierarchical clustering.

Following, the minimality metric in QualIe is based on a three-step approach, which is used for the data values and the schema elements likewise. Consequently, we refer to the observed objects as "elements", using the more generic term for both, records, as well as schema elements.

1) *Element-wise distance calculation.* All elements are compared to each other, which yields a distance matrix.
2) *Clustering.* All elements are hierarchically clustered according to their distance values. In a perfectly minimal IS, the number of elements $|c|$ should be equal to the number of clusters $|clusters|$. If two or more elements are grouped together into one cluster, the minimality score drops to a value below 1.0.
3) *Minimality calculation.* Finally, the minimality can be calculated according to

$$Min(c) = \begin{cases} 1.0, & \text{if } |c| = 1 \\ \frac{|clusters|-1}{|c|-1}, & \text{else} \end{cases}. \tag{9}$$

Schema minimality is of particular interest in the context of IIS, where redundant representations are common. The minimality of a schema is an important indicator to avoid redundancies, anomalies and inconsistencies. QualIe calculates schema minimality according to the three-step approach described above. For the schema similarity, the following distance functions are available: `DSDAttributeDistance` on attribute-level, `DSDConceptAssocDistance` on concept- or association-level, and `SimilarityFloodingDistance` on schema-level. DSD (data source description) is a vocabulary to semantically describe IS schemas [26] and is explained in more detail in Section IV-B. The first two distances are ensemble distances, which are adjusted to the DSD representation of attributes or concepts and associations respectively. In addition, we implemented the Similarity Flooding (SF) algorithm proposed in [27], which calculates the similarity between nodes in a graph-based schema representation, and can thus only be applied to a complete DSD schema (in contrast to single concepts). Subsequently, (9) can be reformulated for schema minimality according to

$$Min(s) = \begin{cases} 1.0, & \text{if } |s| = 1 \\ \frac{|clusters|-1}{|s|-1}, & \text{else} \end{cases}, \tag{10}$$

where $|s|$ is the number of elements (concepts and associations) in a schema s.

E. Normalization

Normal Forms (NFs) can be used to measure the quality of relational DBs, with the aim of obtaining a schema that avoids redundancies and resulting inconsistencies as well as insert, update, and delete anomalies [19]. In contrast to all other schema quality dimensions listed in this paper, normalization requires access to the extension of the information source, i.e., the data values themselves. Although this quality dimension refers to relational data only, it is included in QualIe, because of the wide spread use of relational DBs in enterprises. Several modern DBs use denormalization deliberately to increase read and write performance. Hence, depending on the type of IS, a NF evaluation is not always helpful in deducing the quality of its schema. It can however, serve as checking mechanism to ensure that only controlled denormalization exists.

Identifying *functional dependencies* (FDs) forms the basis for determining the NF of a relation. A FD $\alpha \rightarrow \beta$, where α

and β are two attribute sets of a relation $\mathcal{R}$, describes that two tuples that have the same attribute values in α must also have the same attribute values in β. Thus, the α-values functionally determine the β-values [28].

In QuaIIe, the second, third, and Boyce Codd normal form (2NF, 3NF, and BCNF, respectively) can be determined. The applied algorithm can be classified as a bottom-up method [29], in which the FDs of a relation are analyzed by comparing all attributes' tuple values with all other attributes' tuple values. Then, the minimal cover is determined by performing left- and right-reduction so that all FDs are in canonical form and without redundancies [19]. Following, all attributes are classified as key or non-key attributes and based on all information gathered, the correct NF is determined. Each schema element is annotated with quality information about its NF, key attributes, and minimal cover.

F. Estimation of Integrated Quality Values

In Big Data applications there is usually no gold standard for the entire data set, which makes it impossible to calculate DQ metrics that require a GS in the formula. However, there exist often reference data sets of good quality, for example, purchased customer addresses or a manually cleaned subset of the data. In such cases, DQ can be estimated by extrapolating exact measurements for parts of the data to the entire data set. An estimated quality rating allows to draw conclusions whether to include a data source in a query result or not.

QuaIIe provides a heuristic estimation of DQ values for a number of query results, views, and integrated record sets. Assuming a composite record set can be defined by applying only relational algebra operators (projection π, selection σ, rename ρ, union $\cup$, set difference $-$, and cross product $\times$ [28]) to existing data, queries can be treated as relational syntax trees. From these trees, estimations about the DQ metrics of the composite set can be made without actually evaluating DQ again. Hence, a gold standard is only required for the exact measurement of the leaf components and the DQ estimation for larger (integrated) data is possible without further need of a gold standard [15]. Currently, estimates for the DQ dimensions accuracy, completeness, and pertinence have been implemented in QuaIIe. The DQ metrics of the composite set are estimated by traversing the relational algebra syntax tree in a bottom up fashion utilizing the formulas we present in Tables I and II. Here, $D(c)$ is the proportion of records in a data set c, for which at least one duplicate entry exists in c, and p is a selection-specific factor denoting $\frac{|\text{selected records}|}{|\text{original records}|}$.

IV. Implementation Architecture and Demonstration

Fig. 1 shows the architecture of our modular Java-based tool QuaIIe (pronounced [ˈkvɑlə]) for measuring IIS data and schema quality. The tool consists of three main components: (a) data source connectors to establish an IS connection and load schema information, (b) quality calculators that store information about the schema and data quality in the DQ Store, and (c) reporters to generate a human- and machine-readable quality report. The tool has been implemented with a focus on maximum flexibility and extensibility, which makes it easy to add new connectors, calculators, or reporters, due to a standardized interface for each component. In addition to a pre-configured automatic execution, it also allows user input in form of rules and parameters for specific quality calculations.

In the following paragraphs, each component as well as the DSD Environment and the Data Quality Store are described in more detail and are underpinned with code examples. Fundamentals on the DSD vocabulary are provided in Section IV-B. Recently, a call for more empiricism in DQ research has been proposed in [30], promoting both, (1) the evaluation on synthetic data sets to show the reproducibility of the measurements, and (2) evaluations on large real-world data sets. In this paper, we target the first part since the main contribution is an introduction of QuaIIe and how it can actually be used. We plan to extend the evaluation on real-world data in future work.

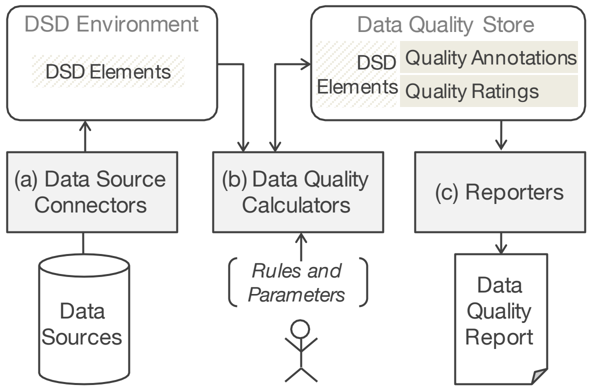

Fig. 1. Implementation Architecture of QuaIIe

A. Demonstration Data Sources

Three different data sources have been employed for this demonstration: employees DB, Sakila DB, and a Comma-Separated Values (CSV) file "Department". We selected those data sources because of their manageable size and well-known qualitative condition, which allows manual tracking and verification of the calculated quality ratings, cf. [30] (in contrast to large real-world data sets with unknown quality).

a) Employees: The employees DB contains six tables with about three million records in total and models the administrations of employees in a company [31]. We employ the `Datasource` object *dsEmpGS* as gold standard for our demonstration, which represents the original employees DB. In addition, we created two variants that have been automatically populated with randomly inserted errors in the original data: *dsEmp1* (501 records in the main table "employees") and *dsEmp2* (4,389 records in the "employees" table). Table III shows the error types that were used in the script. The added noise n is an absolute error that is normally distributed.

TABLE I. DATA QUALITY ESTIMATION - COMPLETENESS AND PERTINENCE

Operator	Composite	Completeness of Composite	Pertinence of Composite
Projection	$\pi(c)$	$Com(c)$	$Per(c)$
Selection	$\sigma(c)$	$p * Com(c)$	$Per(c)$
Union	$c_1 \cup c_2$	$Com(c_1) + Com(c_2) - D(c_1 \cup c_2) * \frac{Com(c_1) + Com(c_2)}{2}$	$\frac{Per(c_1) * \lvert c_1 \rvert + Per(c_2) * \lvert c_2 \rvert}{\lvert c_1 \rvert + \lvert c_2 \rvert}$
Set Difference	$c_1 - c_2$	$Com(c_1) - D(c_1 \cup c_2) * \frac{Com(c_1) + Com(c_2)}{2}$	$\frac{2 * Per(c_1) * \lvert c_1 \rvert - D(c_1 \cup c_2) * (Per(c_1) * \lvert c_1 \rvert + Per(c_2) * \lvert c_2 \rvert)}{2 * \lvert c_1 \rvert - D(c_1 \cup c_2) * (\lvert c_1 \rvert + \lvert c_2 \rvert)}$
Cross Product	$c1 \times c_2$	$Com(c_1) * Com(c_2)$	$Per(c_1) * Per(c_2)$

TABLE II. DATA QUALITY ESTIMATION - ACCURACY

Operator	Composite	Accuracy of Composite
Projection	$\pi(c)$	$Acc(c)$
Selection	$\sigma(c)$	$\frac{Com(c) * p * Acc(c)}{Com(c) * p + (1 - p) * Acc(c)}$
Union	$c_1 \cup c_2$	$\frac{\left(1 - \frac{D(c_1 \cup c_2)}{2}\right) * (Com(c_1) + Com(c_2))}{1 + \left(1 - \frac{D(c_1 \cup c_2)}{2}\right) * \left(Com(c_1) * \left(\frac{1}{Acc(c_2)} - 1\right) + Com(c_2) * \left(\frac{1}{Acc(c_2)} - 1\right) - 1\right)}$
Set Difference	$c_1 - c_2$	$\frac{2 * Com(C_1) - D(c_1 \cup c_2) * (Com(c_1) + Com(c_2))}{2 * \frac{Com(c_1)}{Acc(c_1)} - D(c_1 \cup c_2) * \left(\frac{Com(c_1)}{Acc(c_1)} - \frac{Com(c_2)}{Acc(c_2)}\right)}$
Cross Product	$c1 \times c_2$	$\frac{Com(c_1) * Com(c_2)}{1 + Com(c_1) * \left(\frac{Com(c_2)}{Acc(c_2)} - 1\right) + Com(c_2) * \left(\frac{Com(c_1)}{Acc(c_1)} - 1\right) + \left(\frac{Com(c_1)}{Acc(c_1)} - 1\right) * \left(\frac{Com(c_2)}{Acc(c_2)} - 1\right)}$

TABLE III. ERROR TYPES

Error type	Domain	Example
`LetterSwap`	String	”Bernhard” → ”Bernhrad”
`LetterInsertion`	String	”Bernhard” → ”Bernnhard”
`LetterDeletion`	String	”Bernhard” → ”Bernhrd”
`LetterReplacement`	String	”Bernhard” → ”Burnhard”
`AddedNoise`	Numeric	$a \rightarrow a + n$, where $n \sim N(0, 1)$
`NullFault`	Any	”Bernhard” → NULL
`RecordDuplication`	Record	{(”Werth”, 9)} → {(”Werth”, 9), (”Werth”, 9)}
`RecordDeletion`	Record	{(”Werth”, 9)} → ∅
`RecordInsertion`	Record	{(”Werth”, 9)} → {(”Werth”, 9), (”Ehrlinger”, 5)}
`RecordCrossOver`	Record	{(”Werth”, 9), (”Wöß”, 2)} → {(”Werth”, 2), (”Wöß”, 9)}

b) Sakila: The Sakila DB has 16 tables and models the administration of a film distribution [32]. While the employees DB contains a large number of records for quality measurement on the data-level, Sakila consists of a more advanced schema for schema quality measurement. We employed the `Datasource` object *dsSakilaGS*, which represents the original Sakila DB, as gold standard. In addition, we created *dsSakila1*, *dsSakila2*, and *dsSakila3*, which are excerpts of Sakila including schema modifications to downgrade correctness, completeness, and pertinence respectively.

c) Department CSV: Additionally, a CSV file that contains a list of people affiliated to the department of ”Application-oriented Knowledge Processing” at Johannes Kepler University was used.

As supplement to the demonstration in this paper, we published an executable (`QuaIIe.jar`) on our project website [33], which allows to reconstruct the schema quality measurement described in this section. The program takes one mandatory and one optional command line parameter: (1) the path to the DSD schema to be observed and (2) the path to the gold standard schema, and generates a quality report in XML format. Schema descriptions for all four versions of the Sakila DB, as well as a description for the employees DB are provided in form of DSD files.

B. Data Source Connectors and DSD Environment

A connector’s task is to guarantee data model independence by accessing a data source and transforming its schema into a harmonized schema description, which is based on the the DSD vocabulary. The transformation process from various data models and details of the DSD vocabulary are described in [26]. The transformation from schema elements to DSD elements is a prerequisite for performing cross-schema calculations and obtaining information about a schema’s similarity to other schemas in the IIS. In QuaIIe, DSD elements are represented as dynamically created objects in the Java environment. Below we list the most important terms of the DSD vocabulary that are used in this paper.

- A `Datasource` *s* represents one schema in an IIS and has a type (e.g., relational DB, spreadsheet) and an arbitrary number of concepts and associations, which are also referred to as schema elements.

- A `Concept` c is a real-world object and is usually equivalent to a table in a relational DB or a class in an object-oriented DB.
- An `Association` is a relationship between two or more concepts. There are three types of association: (i) a reference association describes a general relationship between two concepts (e.g., employment of a person with a company); (ii) an inheritance association represents an inheritance hierarchy (e.g., specific types of employees are inherited from a general employee concept); and (iii) an aggregation association describes the composition of several concepts (components) to an aggregate.
- An `Attribute` is a property of a concept or an association; for example, the column "first_name" provides information about the concept "employees".

Fig. 2 shows an example transformation of two relations from the employees DB: `employees {emp_no: int, birth_date: date, first_name: string, last_name: string}` and `dept_emp {emp_no: int, dept_no: int, from_date: date, to_date: date}` into a DSD file in Turtle syntax (cf. [34]). The attribute descriptions are omitted for brevity. The example shows that a relational table can be transformed into a concept or an association, for example, `dept_emp` is a reference association since it models the assignment of an employee to a department.

```
ex:employees a dsd:Concept ;
  rdfs:label "employees" ;
  dsd:hasAttribute ex:employees.emp_no, ex:employees
      .birth_date, ex:employees.first_name, ex:
      employees.last_name;
  dsd:hasPrimaryKey ex:employees.pk .

ex:dept_emp a dsd:ReferenceAssociation ;
  rdfs:label "dept_emp" ;
  dsd:hasAttribute ex:dept_emp.emp_no, ex:dept_emp.
      dept_no, ex:dept_emp.from_date, ex:dept_emp.
      to_date;
  dsd:hasPrimaryKey ex:dept_emp.pk ;
  dsd:referencesTo ex:employees, ex:departments .
```

Fig. 2. Example Schema Description

While this harmonization step enables comparability and quality measurement of schemas from different data models, it does not guarantee access to the original information sources' content after transformation. Consequently, the schema quality metrics in QuaIIe primarily use the schema's metadata instead of the IS content. An exception is the determination of the normal form, which is impossible without considering the semantics of the attributes that can be derived from the content.

There are two different types of connectors in QuaIIe: (1) data source connectors (`DSConnector`), which load the meta data of an IS to describe its schema, and instance connectors (`DSInstanceConnector`), which additionally provide access to the data values of an IS. The interface-oriented design of QuaIIe allows new connectors to be added by implementing one of the two abstract classes `DSConnector` or `DSInstanceConnector`. Currently, three different connectors are supported:

- `ConnectorMySQL` creates a connection to a MySQL DB as representative for relational DBs by using the functionality of the MySQL Java Connector (cf. [35]). This connector allows access to the DB data values. Information on the selected DB schema is retrieved from the data dictionary, including all tables, columns, foreign keys and column properties.
- `ConnectorCSV` allows to access CSV files and is also is a subclass of `DSInstanceConnector`. Due to little meta data available in plain CSV files, schema information is solely extracted from the given file (i.e., column headers as attribute names).
- `ConnectorOntology` uses the Apache Jena framework (cf. [36]) to access a DSD file. Since DSD files hold only schema information and not a connection to the original database, this connector does not provide any possibilities for accessing the DB content and can be used for schema quality measurement only.

Fig. 3 shows an example instantiation for each of the connector types. In addition to opening a connection, it is necessary to load the schema and thus trigger the conversion of schema elements to DSD elements in the Java DSD environment. For our demonstration, we created a connection to all data sources described in Section IV-A, adhering to the same naming standard. For example, for the employees DB we created the connectors *connEmp1* and *connEmp2* to access the MySQL databases with the inserted errors, and load their schema in form of two `Datasource` objects *dsEmp1* and *dsEmp2* into the DSD environment.

```
// Opening and loading a MySQL data source
DSInstanceConnector connEmpGS = ConnectorMySQL.
    getInstance("jdbc:mysql://localhost:3306/", "
    employees", "user", "pw");
Datasource dsEmpGS = connEmpGS.loadSchema();

// Opening and loading a DSD schema description
DSConnector connSakilaGS = new ConnectorOntology("
    filepath/sakila_gs.ttl", "Sakila_Goldstandard");
Datasource dsSakilaGS = connSakilaGS.loadSchema();

// Opening and loading a CSV file
DSInstanceConnector connDept = new ConnectorCSV("
    filepath/department.csv", ",", "\n", "
    FAW_Department");
Datasource dsDept = connDept.loadSchema();
```

Fig. 3. Data Source Connectors

In QuaIIe, each data source connectors also offers at least one gold standard implementation, in order to allow the calculation of reference-based DQ dimensions (e.g., completeness). Fig. 4 shows the creation of two different gold standards: (1) `empGS`, which can be used for quality measurement at the data-level, and (2) `hsGS1`, a `HierarchicalSchemaGS` that is solely used for schema quality measurement. Since specific gold standard implementation might have different tasks, each implementation requires a different set of parameters. However, all gold standards in QuaIIe inherit from the abstract class `GolStandard`, which offers methods to retrieve referenced records or schema elements. The object *empGS* in Fig. 4 shows the instantiation of a gold standard object for a single concept (table), for DQ calculations on different aggregation levels (i.e., when only parts of the content of a data source should be analyzed).

The `HierarchicalSchemaGS` for schema quality calculations extends the idea of simply representing a perfect reference to an information source; rather it is a ”container” that holds the reference to another information source and calculates the similarity or dissimilarity between schema elements on-the-fly. Thus, it is, for example, possible to compare one MySQL DB schema to a DSD description as shown in Fig. 4, to overcome data model heterogeneity.

```
// Creation of a gold standard from a single concept
GoldStandard empGS = new StrictConceptMySQLGS(
    dsEmpGS.getConcept("employees"), connEmpGS);

// Creation of a schema gold standard
GoldStandard hsGS1 = new HierarchicalSchemaGS(
    dsSakila1, dsSakilaGS);
```

Fig. 4. Gold Standards

C. Data Quality Calculators and DQ Store

Each DQ calculator is dedicated to one of the quality dimensions described in Section III and links the measurements to the corresponding DSD elements in the DQ store. Quality measurements in the DQ store can be used for reporting or reused by other calculators, and can be divided into two different types: *quality ratings* or *quality annotations*. A rating is a double value between 0.0 and 1.0, which is calculated by a specific metric that is assigned to a quality dimension. An example would be a value of 0.85 for the dimension ”completeness” on data-level using the metric ”ratio”. A quality annotation can be an arbitrary object that is linked to a DSD element in the DQ store in order to provide additional information about the quality. An example would be the annotation of functional dependencies to a concept.

Fig. 5 shows the application of all non-time-related DQ calculators that are implemented in the current version of QuaIIe. Initially, the concept ”employees” from the erroneous `Datasource` *dsEmp1* is selected for closer investigation. As an example for a distance function, which is required for the minimality calculation, line 5-7 cover the creation of an `EnsembleDistance`, which is a weighted combination of an arbitrary number of specific distance functions. In the demo, we use a combination of two string distances for the attributes `first_name` and `last_name` in the ”employees” table. However, QuaIIe allows the creation of arbitrary complex distance functions for each record. Finally, ratings for the DQ dimensions accuracy, completeness, pertinence, and minimality are calculated. Line 16 shows how to programmatically retrieve those stored DQ values from the DQ store. One data quality rating or annotation is uniquely identifiable in the DQ store by a reference to the DSD element (e.g., a reference to the concept ”employees” in *dsEmp1*), the `DIMENSION_LABEL` of the measured quality dimension (e.g., ”completeness”) as well as a `METRIC_LABEL` (e.g., ”ratio”), which describes the metric used for calculating the dimension.

```
1  // Select concept "employees" from employees DB
2  Concept c = dsEmp1.getConcept("employees");
3
4  // Create a custom distance measure
5  EnsembleDistance<Record> dist = new EnsembleDistance
       <Record>();
6  dist.addDistance(new StringRecordDistance(c.
       getAttribute("first_name"), new
       LevenshteinDistance()), 0.5);
7  dist.addDistance(new StringRecordDistance(c.
       getAttribute("last_name"), new
       LevenshteinDistance()), 0.5);
8
9  // Perform quality calculations
10 RatioAccuracyCalculator.calculate(c,empGS,connEmp1);
11 RatioCompletenessCalculator.calculate(c,empGS,
       connEmp1);
12 RatioPertinenceCalculator.calculate(c, empGS,
       connEmp1);
13 RecordMinimalityCalculator.calculate(c, dist, 0.1,
       connEmp1);
14
15 // Retrieve DQ measurements from the DQ store
16 DataQualityStore.getDQValue(c,
       RatioPertinenceCalculator.DIMENSION_LABEL,
       RatioPertinenceCalculator.METRIC_LABEL)
```

Fig. 5. Data Quality Calculations

In addition to the measurement of *dsEmp1* (501 records), we applied the same calculations on the ”employees” table of *dsEmp2* (4,389 records). The results can be compared in Table IV. The low quality values for accuracy and completeness are due to the small subsets of the erroneous tables in contrast to the original employees table with 30,0024 records.

TABLE IV. DQ MEASUREMENT OF ERRONEOUS DATA SOURCES

Dimension	*dsEmp1*	*dsEmp1*
Accuracy	0.0013	0.0116
Completeness	0.0013	0.0116
Pertinence	0.7725	0.7938
Minimality	0.7180	0.7532

For the schema quality calculations, we employed a DSD description of the original Sakila DB as gold standard and accessed the three additional data sources (*dsSakila1*, *dsSakila2*, *dsSakila3*) through the MySQL connector. Each data source contains schema modifications that tackle one of the schema quality dimensions correctness, completeness, and pertinence, and are justified in the following paragraphs. For the demonstration using `QuaIIe.jar` on our project website [33], we provided all four schemas as DSD files in order to facilitate data exchange and reproduction.

The 16 tables from Sakila were transformed into 14 DSD concepts and two DSD reference associations (`film_category` and `film_actor`). For the *hierarchical schema similarity*, standard parameters have been used with a less restrictive attribute similarity threshold of 0.8. The determination and evaluation of the schema similarity standard parameters is justified in [14]. Fig. 6 shows the application of the schema quality calculators.

```
HierarchicalSchemaCorrectness.calculate(dsSakila1,
    hsGS1);
HierarchicalSchemaCompleteness.calculate(dsSakila2,
    hsGS2);
HierarchicalSchemaPertinence.calculate(dsSakila3,
    hsGS3);
RatioMinimalityCalculator.calculate(dsEmpGS);
NormalFormCalculator.calculate(dsEmpGS, connEmpGS);
```

Fig. 6. Schema Quality Calculations

The schema quality measurements that have been generated in Fig. 6, are 0.8125 for correctness, 0.8125 for completeness, 0.8824 for pertinence, and 0.8 for minimality. The results are discussed in more detail in the following subsections.

a) Schema Correctness: In order to demonstrate the correctness dimension, we performed changes in the observed schema but did not remove or add new schema elements. The corresponding DQ report can be generated by executing `java -jar QuaIIe.jar sakila_correctness.ttl sakila_gs.ttl`. First, the concept `film` was renamed to ”movie”, which did not change the ratings for pertinence and completeness, but decreased correctness slightly to 0.9375 due to the additional incorrect element. Second, all occurrences of film (e.g., `film_id`) in the DB were replaced with ”movie”. While completeness and pertinence retained a rating of 1.0, because all concepts and associations were assigned (even if incorrectly) to their original correspondences in the GS, correctness achieved only a rating of $\frac{13}{13+3+0} = 0.8125$.

b) Schema Completeness: The completeness calculation was demonstrated by removing schema elements. The DQ report for this demo can be generated by assessing `sakila_completeness.ttl`. Initially, the two tables `category` and `film_category` were removed, which resulted in a completeness rating of $\frac{14+0}{14+0+2} = 0.875$ because two elements were classified as missing. Then, the attribute `picture` was deleted from the table `staff`. Removals at the attribute level did not directly affect the result of the completeness calculation, since `staff` is still correctly assigned to its gold standard representation due to the tolerance of the distance calculation. Concluding, three additional attributes were removed from `staff`, which resulted in a similarity rating of 0.6923 between `staff` and its correspondence in the GS. Consequently, both tables were not mapped because they were too different and completeness dropped to $\frac{13+0}{13+0+3} = 0.8125$.

c) Schema Pertinence: For the demonstration of pertinence, we added additional elements to the schema and the quality report can be generated by assessing the file `sakila_pertinence.ttl`. In a first step, the ”employees” table from the employees DB was added to *dsSakila3*, dropping pertinence to 0.9412. This demo correctly classifies the concept `employees` as an extra element, although the new concept has a relatively low distance to the concept `actor`. Second, we modified the concept `actor` in *dsSakila3*, such that no assignment to its corresponding concept in the GS was created and the pertinence rating dropped to $\frac{15+0}{15+0+2} = 0.8824$. Following, the newly added `employees` table was aligned with the `actor` concept in the GS by removing and altering attributes. This resulted in a similarity value of 0.8333 between `employees` and the concept `actor` from the GS and increased completeness to 1.0 (all elements could be assigned to the GS). However, the pertinence dimension (0.9412) indicated the extra `actor` concept in the observed schema, which did not match any of the GS elements.

We conclude that an examination of all three dimensions (correctness, completeness, and pertinence) is advisable when measuring the quality of a schema. Note that the correctness metric is particularly strict, because it is decreased by every incorrect element in the schema, whereas completeness and pertinence do not distinguish between correct and incorrect.

d) Schema Minimality: Analogous to the data minimality, schema minimality requires a distance function. Currently, two schema distance functions are offered: the similarity flooding algorithm introduced in [27] and hierarchical schema similarity, which we use in the following calculations with standard parameters that have been evaluated in [14]. First, we observed the Sakila DB schema (`sakila_gs.ttl`), which achieves an ideal minimality rating of $\frac{16-1}{16-1} = 1.0$, because all schema elements are sufficiently different to each other.

Second, we evaluated the employees schema, which yields the similarity matrix in Table V. Interestingly, the two associations `dept_emp` and `dept_manager` achieve a very high similarity of 0.875, which reduces the minimality rating to $\frac{5-1}{6-1} = 0.8$. In practice, this rating indicates an IS architect that the two associations should be further analyzed. However, in our case, no further action is required since the employees schema contains a special modeling concept of parallel associations (i.e., two different roles) which does not represent semantic redundancy, but leads to very similar relations in the schema model (cf. [31]). Since it is known that this modeling construct yields high similarity values (e.g., also for schema matching applications), it was specially suited to demonstrate our minimality metric. The full quality report for this demo can be generated by executing ”`java -jar QuaIIe.jar employees.ttl`”.

TABLE V. SIMILARITY MATRIX FOR EMPLOYEES SCHEMA

	depts*	dept_emp	dept_mgr*	employees	salaries	titles
depts*	1.0	0.125	0.125	0.1	0.125	0.125
dept_emp	0.125	1.0	0.875	0.1818	0.2222	0.1
dept_mgr*	0.125	0.875	1.0	0.1818	0.2222	0.1
employees	0.1	0.1818	0.1818	1.0	0.1818	0.1818
salaries	0.125	0.2222	0.2222	0.1818	1.0	0.375
titles	0.125	0.1	0.1	0.1818	0.375	1.0

*Departments is abbreviated with “depts” and dept_manager with “dept_mgr”.

e) Normal Form Calculation: The NF calculator was applied to the employees DB and yields BCNF for each concept. The minimal cover of the FDs is shown in Table VI.

Due to the large number of records in the employees database, calculating these results took about 22 minutes and 45 seconds on a Macbook Pro with an Intel Core i7 processor with 2,2 GHz and 16 GB main memory. In addition to FDs, candidate keys are also annotated to the observed schema elements, and attributes are annotated with a Boolean value that indicates whether they are classified as key or non-key. Note that, particularly in terms of performance, more sophisticated methods of discovering FDs exist [29]. However, since the main aim of our work was to provide a comprehensive approach to data and schema quality measurement, the normalization dimension was included for completeness to support full FD discovery (i.e., without approximation).

TABLE VI. NF CALCULATION - EMPLOYEES SCHEMA

Concept	Functional Dependencies
departments	{dept_no}→{dept_name}, {dept_name}→{dept_no}
dept_emp	{emp_no, dept_no}→{from_date, to_date}
dept_manager	{emp_no}→{dept_no, from_date, to_date}
employees	{emp_no}→{first_name, last_name, gender, birth_date, hire_date}
salaries	{emp_no, from_date}→{to_date, salary}
titles	{emp_no, title, from_date}→{to_date}

D. Data Source Integration

In IIS, it is often necessary to estimate the quality of data stemming from different IS. QuaIIe supports the virtual integration of different concepts, which is realized with the Java classes `IntegratedDatasource` and `IntegratedConcept`. Fig. 7 shows an example integration, where all records from the table ”employees”, which is present in both erroneous data sources *dsEmp1* and *dsEmp2*, are unified. The data is stored in form of a virtual integrated data source (`ids`), which exists only during runtime.

```java
IntegratedDatasource ids = DSDFactory.
    makeIntegratedDatasource("integratedEmp");

ISQLIntegrator integrator = new ISQLIntegrator(ids);
integrator.add(dsEmp1, connEmp1);
integrator.add(dsEmp2, connEmp2);

IntegratedConcept ic = integrator.
    makeIntegratedConceptFromString("SELECT * FROM
    dsEmp1.employees UNION SELECT * FROM dsEmp2.
    employees", "integratedEmployees");
```

Fig. 7. Data Integration

An integrated concept contains an *operator tree*, which specifies the data sources, concepts, connectors, and integration transformations that are required for its creation. After generating such an integrated concept, it can be assessed likewise to an ordinary concept from a single data source in QuaIIe (cf. lines 3-6 in Fig. 8). Additionally, it is possible to estimate the quality (cf. Section III-F), which is not a complete measurement of the new integrated concept, but is based on the prior quality ratings of each IS. Thus, an estimation requires the prior measurement of each IS that takes part in the integration.

```java
1  DSInstanceConnector integrConn = new
       IntegratedInstanceConnector(ic);
2
3  RatioCompletenessCalculator.calculate(ic, gsEmp,
       integrConn);
4  RatioAccuracyCalculator.calculate(ic, gsEmp,
       integrConn);
5  RatioPertinenceCalculator.calculate(ic, gsEmp,
       integrConn);
6  RecordMinimalityCalculator.calculate(ic, dist, 0.1,
       integrConn);
7
8  RatioCompletenessCalculator.estimate(ic);
9  RatioAccuracyCalculator.estimate(ic);
10 RatioPertinenceCalculator.estimate(ic);
```

Fig. 8. DQ Estimation of an integrated Concept

The ratings for the DQ calculations and estimations from Fig. 8 are compared in Table VII and show high conformance. In the current version of QuaIIe, quality estimation is only available for the dimensions accuracy, completeness, and pertinence. However, an extension of the DQ estimators to other dimensions, like minimality, is planned as future work.

TABLE VII. DQ CALCULATION OF AN INTEGRATED CONCEPT

Dimension	Measurement	Estimation
Accuracy	0.0129	0.0130
Completeness	0.0129	0.0128
Pertinence	0.7916	0.7916
Minimality	0.7494	-

E. Data Quality Report

In order to present the quality ratings and annotations contained in the DQ store in a human- and machine-readable way, QuaIIe offers several reporter classes that generate a quality report. The most comprehensible end-user report is `XMLTreeStructureDQReporter`, which is created in Fig. 9 and exports a description of all connected data sources with their DSD elements, quality ratings and annotations. Since such a report tends to be large and verbose for large IIS, the hierarchical structure of the XML document allows to drill-down and roll-up on different aggregation levels by using a suitable viewer. In addition, languages like XSLT, XQuery, or XPath allow a user to search within such a report. The advantage of an XML report in our use case is however the fact that it can be reused automatically for further analysis and benchmarking (e.g., for data quality monitoring). When measuring the quality of the published DSD schemas with `QuaIIe.jar` (cf. [33]), the output is such a report.

```java
XMLTreeStructureDQReporter reporter = new
    XMLTreeStructureDQReporter();
reporter.buildReport();
reporter.writeReport("path/DQReport.xml");
```

Fig. 9. Data Quality Report Generation

V. Conclusion and Future Work

In this paper, we have described QuaIIe, a tool to estimate and measure the data and schema quality of an IIS.

QuaIIe generates a machine- and human-readable quality report, which allows for repetitive quality measurement and comparison of different reports from the same IIS. The DQ measurement covers the dimensions accuracy, correctness, completeness, pertinence, minimality, and normalization for both, the schema and the data of an IS. To the best of our knowledge, there exists no tool that measures such a large number of different DQ dimensions in a single application. However, our major contribution is the unsupervised and automated ad-hoc analysis of both data and schema quality.

Our ongoing and future work focuses on a practical evaluation of QuaIIe on real-world data with respect to the benefits of the measured DQ metrics. Coupled to the evaluation, we plan to extend the theoretical foundations by more deeply considering research from related fields, like data cleansing and integration. In addition, several implementation extensions like a calculator for the readability dimension as well as a connector for Oracle DBs and a connector for Apache Cassandra are currently under development. An implementation of a graphical user interface to visualize the DQ measurements is also planned.

Acknowledgment

The research reported in this paper has been partly supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

References

[1] KPMG International, “Now or Never: 2016 Global CEO Outlook,” 2016.

[2] T. C. Redman, “The Impact of Poor Data Quality on the Typical Enterprise,” *Communications of the ACM*, vol. 41, no. 2, pp. 79–82, Feb. 1998.

[3] ——, “Bad Data Costs the U.S. $3 Trillion Per Year,” Harvard Business Review, 2016, https://hbr.org/2016/09/bad-data-costs-the-u-s-3-trillion-per-year [retrieved: March, 2018].

[4] B. Otto and H. Österle, *Corporate Data Quality: Prerequisite for Successful Business Models*. Berlin, Germany: Berlin: Springer Gabler, 2016.

[5] F. Naumann, U. Leser, and J. C. Freytag, “Quality-driven Integration of Heterogenous Information Systems,” in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB ’99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 447–458.

[6] Y. Wand and R. Y. Wang, “Anchoring Data Quality Dimensions in Ontological Foundations,” *Communications of the ACM*, vol. 39, no. 11, pp. 86–95, Nov. 1996.

[7] M. Y. Selvege, S. Judah, and A. Jain, “Magic Quadrant for Data Quality Tools,” Gartner, Tech. Rep., October 2017.

[8] J. Barateiro and H. Galhardas, “A Survey of Data Quality Tools,” *Datenbank-Spektrum*, vol. 14, no. 15-21, p. 48, 2005.

[9] V. Pushkarev, H. Neumann, C. Varol, and J. R. Talburt, “An Overview of Open Source Data Quality Tools,” in *Proceedings of the 2010 International Conference on Information & Knowledge Engineering, IKE 2010, July 12-15, 2010, Las Vegas Nevada, USA*, 2010, pp. 370–376.

[10] R. Y. Wang and D. M. Strong, “Beyond Accuracy: What Data Quality Means to Data Consumers,” *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5–33, Mar. 1996.

[11] C. Batini and M. Scannapieco, *Data and Information Quality: Concepts, Methodologies and Techniques*. Springer International Publishing, 2016.

[12] “Standard for a Software Quality Metrics Methodology,” Institute of Electrical and Electronics Engineers, IEEE 1061-1998, 1998.

[13] Oxford University Press, “Definition of Gold Standard in English,” Online, 2017, http://www.oxforddictionaries.com/definition/american-_english/gold-standard [retrieved: March, 2018].

[14] L. Ehrlinger, “Data Quality Assessment on Schema-Level for Integrated Information Systems,” Master’s thesis, Johannes Kepler University Linz, 2016.

[15] B. Werth, “Identifikation von Datenqualitätsproblemen in integrierten Informationssystemen [Identification of Data Quality Issues in Integrated Information Systems],” Master’s thesis, Johannes Kepler University Linz, 2016.

[16] T. Haegemans, M. Snoeck, and W. Lemahieu, “Towards a Precise Definition of Data Accuracy and a Justification for its Measure,” in *Proceedings of the International Conference on Information Quality (ICIQ)*, 2016, pp. 16:1–16:13.

[17] J. R. Logan, P. N. Gorman, and B. Middleton, “Measuring the Quality of Medical Records: A Method for Comparing Completeness and Correctness of Clinical Encounter Data,” in *AMIA 2001, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 3-7, 2001*, 2001, pp. 408–4012.

[18] G. Salton and M. J. McGill, “Introduction to Modern Information Retrieval,” 1986.

[19] G. Vossen, *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme [Data Models, Database Languages, and Database Management Systems]*. Oldenbourg Verlag, 2008.

[20] M. C. M. Batista and A. C. Salgado, “Information Quality Measurement in Data Integration Schemas,” in *Proceedings of the Fifth International Workshop on Quality in Databases, QDB 2007, at the VLDB 2007 Conference, Vienna, Austria*. ACM, September 2007, pp. 61–72.

[21] F. Naumann, J.-C. Freytag, and U. Leser, “Completeness of Integrated Information Sources,” *Information Systems*, vol. 29, no. 7, pp. 583–615, Sep. 2004.

[22] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate Record Detection: A Survey,” *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[23] I. P. Fellegi and A. B. Sunter, “A Theory for Record Linkage,” *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[24] J. Euzenat and P. Shvaiko, *Ontology Matching*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[25] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering: A Review,” *ACM Computing Surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 2000.

[26] L. Ehrlinger and W. Wöß, “Semi-Automatically Generated Hybrid Ontologies for Information Integration,” in *Joint Proceedings of the Posters and Demos Track of 11th International Conference on Semantic Systems – SEMANTiCS2015 and 1st Workshop on Data Science: Methods, Technology and Applications (DSci15)*. CEUR Workshop Proceedings, 2015, pp. 100–104.

[27] S. Melnik, H. Garcia-Molina, and E. Rahm, “Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching,” in *Proceedings of the 18th International Conference on Data Engineering*, ser. ICDE ’02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 117–128.

[28] E. F. Codd, “A Relational Model of Data for Large Shared Data Banks,” *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[29] J. Liu, J. Li, C. Liu, and Y. Chen, “Discover Dependencies from Data – A Review,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 2, pp. 251–264, 2012.

[30] S. Sadiq, T. Dasu, X. L. Dong, J. Freire, I. F. Ilyas, S. Link, M. J. Miller, F. Naumann, X. Zhou, and D. Srivastava, “Data Quality: The Role of Empiricism,” *ACM SIGMOD Record*, vol. 46, no. 4, pp. 35–43, 2018.

[31] Oracle Corporation, “Employees Sample Database,” Online, https://dev.mysql.com/doc/employee/en [retrieved: March, 2018].

[32] ——, “Sakila Sample Database,” Online, https://dev.mysql.com/doc/sakila/en [retrieved: March, 2018].

[33] L. Ehrlinger, “Data Quality Assessment for Heterogenous Information Systems,” Online, http://dqm.faw.jku.at [retrieved: March, 2018].

[34] W3C Working Group, “RDF 1.1 Turtle,” Online, 2014, https://www.w3.org/TR/turtle [retrieved: March, 2018].

[35] Oracle Corporation, “MySQL Connectors,” Online, https://www.mysql.com/products/connector [retrieved: March, 2018].

[36] The Apache Software Foundation, “Apache Jena,” Online, https://jena.apache.org [retrieved: March, 2018].