Automated Continuous Data Quality Measurement with QuaIIe

Lisa Ehrlinger*†
†Software Competence Center Hagenberg GmbH
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@jku.at, bernhard.werth@fh-hagenberg.at, wolfram.woess@jku.at

***Abstract*—Data quality measurement is essential to gain knowledge about data used for decision-making and to evaluate the trustworthiness of those decisions. Example applications, which are based on automated decision-making, are self-driving cars, smart factories, and weather forecast. One-time data quality measurement is an important starting point for any data quality project to detect critical data that does not meet expectations and to define improvement goals for data cleansing activities. The complementary task of *continuous* data quality measurement is essential to ensure that data continues to conform to requirements and to detect unexpected changes in the data. However, most existing data quality tools allow quality measurement at a specific point in time while leaving the automation and scheduling to the user. In this paper, we highlight the need for (1) domain-independent ad hoc measurement, to provide a quick insight of an information system's qualitative condition, and (2) continuous data quality measurement, to observe how data quality evolves over time. Both requirements can be achieved with our data quality tool QuaIIe (Quality Assessment for Integrated Information Environments, pronounced ['kvɑlə]), which we developed to calculate metrics for the quality dimensions accuracy, correctness, completeness, pertinence, timeliness, minimality, readability, and normalization on both data-level and schema-level. The quality measurements can be either exported as a user- and machine-readable quality report, or they can be periodically stored in a database, which allows for long-term analysis. In this paper, we demonstrate the application of QuaIIe for ad hoc and continuous data quality measurement.**

***Index Terms*—Data Quality; Measurement; Monitoring; Estimation; Trust.**

I. Introduction

Strategic decisions are usually based on data. Examples are human-made decisions in enterprises whether to promote or to suspend the production of a specific product due to sales data. In the era of artificial intelligence, machine-based decisions gain increasing relevance in applications like self-driving cars, search engines, or industry robots. In order to trust such data-driven decisions, it is necessary to measure and know the quality of the underlying data with appropriate tools [1]. Despite the clear correlation between data and decision quality, 84 % of the CEOs in the US are concerned about their data quality [2]. In addition to incorrect decision making, poor Data Quality (DQ) may cause effects like cost increase, customer dissatisfaction, and organizational mistrust [3]. According to an estimation by IBM, the total financial impact of poor quality data on business in the US was $3.1 trillion [4] 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 [5].

We originally presented the Java-based DQ tool QuaIIe in [1], which automatically performs domain-independent quality measurement on both data-level and schema-level. The name "QuaIIe" is not only an abbreviation for "Quality Assessment for Integrated Information Environments", but it is also the German word for jellyfish. We consider this amazing but yet not fully explored animal a proper representative for our DQ tool, since its complex life cycle consists of two major stages, which are divided by two different reproduction phases: (1) the stationary polyp-phase, and (2) the free-swimming medusa-phase [6]. In analogy to the jellyfish life cycle, the two major aims of QuaIIe are:

(1) Initial ad hoc data quality measurement (*stationary*)
(2) Continuous data quality measurement (*free-swimming*)

Both phases are necessary to provide holistic DQ measurement. In the stationary polyp-phase, a machine- and human-readable XML (extensible markup language) quality report is generated, which allows to grasp a first insight into the qualitative condition of an information system (IS) without requiring preparation activities or deeper domain-knowledge. Initial ad hoc DQ measurement guides the path for more detailed investigation and target-oriented data profiling. The medusa-phase allows the long-term observation of an IS's quality in order to detect trends or outliers over time. Continuous DQ measurement provides indications for further DQ improvements and thus, increases the trustworthiness for data-driven decisions. While the polyp-phase was originally introduced in [1], the medusa-phase is the major contribution of this paper. In addition, we present several implementation extensions of QuaIIe: a new data source connector for Apache Cassandra [7], new metrics for the DQ dimensions timeliness and readability, and experiments with large and highly volatile real-world data.

Jellyfish also contribute to science with their green fluorescent protein (GFP), which can be used as marker for gene expressions or as reporter for virus infections [8]. QuaIIe marks DQ issues for the quality dimensions accuracy, correctness, completeness, pertinence, timeliness, minimality, readability, and normalization. Since data of enterprises and organizations are usually stored in Integrated Information Systems (IISs) [9], an important feature of a DQ tool is to estimate the quality of different and often heterogeneous ISs on-the-fly to select the most appropriate and most trustworthy source for a given query. The jellyfish tentacles of QuaIIe allow virtual integration of different data sources or parts of data sources [1].

While a number of DQ tools has been proposed over the years (cf. [10][11][12][13]), most domain-independent tools focus on data cleansing and/or data integration. We want to highlight that our focus is the quality measurement of an IIS in productive use, and automatic data cleansing activities are not in the scope of this research work. In order to understand the degree and effectiveness of data cleansing and to define goals for further cleansing activities, it is necessary to measure and know the quality of the data [14]. To the best of our knowledge, there exists no tool that offers (continuous and ad hoc) DQ measurement for such a large number of different DQ dimensions in a single application and comprises both data and schema quality. Thus, the main contributions of QuaIIe in contrast to other DQ tools can be summarized as follows:

- Domain-independent ad hoc DQ measurement
- Continuous data quality measurement
- Virtual data integration as basis for DQ estimation
- Combination of data and schema quality measurement

The remainder of this paper is organized as follows: in Section II, we discuss the state-of-the-art into data quality, clarify the concept of "continuous data quality measurement" and differentiate QuaIIe from existing DQ tools. Section III covers all data and schema quality dimensions and metrics, which were applied in this research. In Section IV, we describe the implementation of QuaIIe and demonstrate its application in terms of ad hoc and continuous DQ measurement.

II. Data Quality: State of the Art

Data quality is usually defined by the "fitness for use" principle [15][16][17], which refers to the high subjectivity and context-dependency of this topic. This definitions emphasizes the fact that DQ cannot be evaluated without considering the context (e.g., a specific application or service), in which the data is used for. To grasp single qualitative aspects on the data, DQ is typically described by a set of DQ *dimensions* (e.g., accuracy, completeness, timeliness) and *metrics*, which are specific formulas to calculate the dimensions [15][18]. A methodology for DQ can be divided into the following activities [19][20]: (1) state reconstruction, (2) DQ measurement, (3) data cleansing or improvement, and (4) continuous DQ measurement or monitoring. Step (1) comprises the collection of contextual information on the observed data and the organization where a DQ project is carried out [19]. DQ measurement (2) is typically described by a set of DQ dimensions and assigned metrics. Until now, there exists no agreement on a standardized set of dimensions and metrics for DQ measurement [14]. Data cleansing (3) is the process of correcting erroneous data and includes tasks like customer data standardization or data de-duplication. According to [20], automated data cleansing methods are very valuable for Big Data, but with the risk to insert new errors. Step (4) is mainly used implicitly in literature in order to refer to the ongoing observation of an IS's quality and is discussed in more detail in Section II-A. While we use the term "continuous DQ measurement" to describe this DQ activity, synonymously used terms are "recurrent DQ assessment" [20], "ongoing DQ measurement" [14], or "automated DQ monitoring" [21]. The importance for *automation* in data quality projects is highlighted in [14], for both, initial and ongoing DQ measurement. Without automation, the volume and volatility of data in complex real-world IS will quickly overwhelm any DQ measurement efforts [14].

A. Continuous Data Quality Measurement

The term "continuous data quality measurement" (CDQM) describes the calculation and storage of DQ metrics over time, in order to ensure that the qualitative condition of the data remains stable [22]. Sebastian-Coleman [14] distinguishes between in-line measurement, periodic measurement, and controls. In-line and periodic measurement are distinguished by the measurement frequency, where in-line measurement is applied to critical or highly volatile data, and periodic measurement is used to monitor less frequently updated data (e.g., master data) [14]. Our implementation QuaIIe allows to apply both types: in-line as well as periodic DQ measurement. Table I describes the most important requirements for CDQM defined in [22] and how they are implemented in QuaIIe.

A control is understood as a built-in data check, which is typically implemented in a data transformation (or integration) process [14]. We want to point out that the majority of DQ tools that state to offer *monitoring* functionality refer to controls. Also Gartner [10] describe the term "monitoring" as "the deployment of controls in order to ensure that data continues to conform to business rules". We explicitly want to distinguish our understanding of CDQM (i.e., in-line and periodic measurement), as it is implemented in QuaIIe, from DQ monitoring using controls. A practical example of a monitoring tool with controls in the physics domain is the DQ monitoring framework for the ATLAS experiment at the Large Hadron Collider at CERN [27]. Before the automatically collected data is shipped to the data store, a number of predefined quality checks are carried out in order to ensure that the data is free of error and can be used for scientific data analysis. This DQ tool is accompanied by human domain experts, who inspect the generated visualization and take action in case of problems.

In accordance with [14], we want to highlight that the usefulness and application of CDQM depends on the volatility of the data set. We distinguish between three volatility types:

(V1) *Static data sets*, which are very rarely or not modified, for example, a list of planets in the solar system or countries and their capitals.

(V2) Periodically or *occasionally updated data*, for example, master data like products, customers, employees, or the daily menu of a restaurant.

(V3) *Highly volatile data*, for example, stock exchange prices, sales data from large vendors, streaming or sensor data.

All three types require different measurement strategies, which are demonstrated with QuaIIe in Section IV. The

TABLE I
REQUIREMENTS FOR CONTINUOUS DATA QUALITY MEASUREMENT

Requirement	Description	Implementation in QuaIIe
DQ measurement functionality	To define how DQ should be actually measured is not trivial and according to [14], one of the biggest challenges for DQ practitioners. One reason is the ongoing discussion on DQ dimensions, where until now, no agreement could be reached [14].	QuaIIe currently implements metrics for eight different DQ dimensions on both data-level and schema-level. Since not all published DQ metrics have been sufficiently evaluated so far (cf. [23]), QuaIIe allows to evaluate metrics, e.g., whether they are suited for CDQM or not.
Storage	DQ measurements must be persisted to allow DQ analysis over time and to compare current DQ measurements to older ones.	For the experiments in this paper, we used a MySQL database to store CDQM results, but any kind of database might be used.
Automation	DQ measurement should be performed automatically, based on user-defined time periods.	We automated DQ measurements either directly in a Java method, or with Windows Task Scheduler [24].
Analysis	Visual as well as statistical (time-series) analysis is required to present the DQ measurement results and to detect patterns and changes in DQ measurements.	Since a graphical user interface for QuaIIe is currently under development, we performed the analysis of CDQM with the Python packages pandas [25] and matplotlib [26].

strategies differ in the measurement frequency (in-line versus periodic), in the amount of data to be included in the measurement, as well as in the automatically triggered action in case a specific threshold is not met.

B. Data Quality Tools

Despite a continuous growth, the market of DQ tools is still considered a niche market [10]. Gartner lists 39 commercial DQ tools by 16 vendors in their "Magic Quadrant of Data Quality Tools 2017" [10]. Most of the tools offer functionalities to investigate the qualitative condition of different data sources using data profiling techniques, 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 have been proposed over the years, where the most important ones are compared and discussed in [11][12][13]. All three 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) and none of the observed tools provide any monitoring functionality. Examples for typical data cleansing tools are Potter's Wheel [28] or Wrangler [29]. In contrast to the tools observed in existing surveys [11][12][13], QuaIIe focuses on the pure measurement (detection) of DQ problems and does not cleanse data. The advantage of DQ measurement without cleansing or manipulating data is the potential for domain-independent automation, that is, it can be performed unsupervised and ad hoc without any consequences to insert new errors in the data.

Additionally, and in contrast to most existing DQ tools (except for [30] and [31], which will be discussed in the following paragraph) QuaIIe addresses the DQ topic from the dimension-oriented view. While a lot of research on DQ dimensions and their definitions has been proposed in literature [3][15][17], there is no tool that implements generally-applicable 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. Of course, more specialized tools might outperform QuaIIe in specific implementations, like distance calculation or string matching.

In terms of CDQM capabilities and the DQ dimension-oriented view, we found two open source tools that can be compared to QuaIIe: MobyDQ by Alexis Rolland [30] (formerly: "Data Quality Framework") and Apache Griffin [31], a project from the Apache Incubator. However, both tools require an intensive configuration phase, where DQ metrics and checks are defined depending on the observed domain and data. No metrics for initial ad hoc measurement are provided. While MobyDQ is easy to install, Apache Griffin is very arduous to set up, because it depends on several other open source tools that are still in the incubator status. We needed six days for the installation until we were able to produce usable DQ measurements.

III. Data and Schema Quality Dimensions

Data quality is usually described as multidimensional concept, which is characterized by different aspects, so called *dimensions* [15]. 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) [18]. While the majority of research into DQ focuses on the data values, QuaIIe implements DQ measurements for both schema and data values. In fact, schema quality has a strong impact on the quality of the data values [18]. 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 (e.g., [15][17][32][33]), we focus in the following paragraphs on the eight dimensions (1) accuracy, (2) correctness, (3) completeness, (4) pertinence, (5) timeliness, (6) minimality, (7) readability, and (8) normalization. Each dimension can be quantified with one or several metrics, which capture the fulfillment of a dimension in a numerical value [34]. Heinrich et al. [23] defined five requirements a

data quality metric must fulfill. The implemented metrics in QuaIIe can thus be evaluated by means of the requirements in [23] or if they are suited for application in CDQM.

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" [35]. 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-H) 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 [36] and more information on the data quality dimensions can be found in [37].

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 [15][18][38]. 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 [18]. From the natural sciences perspective, accuracy is usually defined as the magnitude of an error [38]. In this research work, we refer to correctness for a calculation, which has been presented by Logan et al. [39], who distinguish between correct (C), incorrect (I), extra (E) and missing (M) elements after comparing a data set to its reference:

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

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 show below that metrics for correctness and pertinence can be determined by the same basic element counts (C, I, E, M – short CIEM), which allows an efficient implementation. Algorithm 1 demonstrates how the element counts are calculated.

On the data-level, QuaIIe implements an accuracy metric, which has its origins in the field of machine learning and is usually used to measure the accuracy of classification algorithms [40]. This accuracy metric can also be mapped to the notion by Logan et al. [39]:

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

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 [41] describes a database (DB) schema as correct, if the concepts of the related data model are applied in a syntactically and semantically appropriate way, effectively considering only the model as reference. In [18] the authors distinguish between correctness with respect to the model and with respect to requirements. Although the values of an IS might also be erroneous, the content of an IS can be added as third possibility to validate a schema. Three types of validation are therefore possible:

- Validation of a schema against its conceptual *model* (e.g., Entity-Relationship model) assumes the correct representation of the modeled constructs within the schema.
- When a schema is validated against its *requirements*, the requirements are expected to be represented correctly. This is usually considered to be a manual task (cf. [42]), because requirements are rarely available in machine-readable form.
- Validation against the *content* of an information source verifies whether the schema fits its values. This includes for instance the correct usage of attributes (e.g., an

Algorithm 1: Calculation of CIEM Counts

Input: Data set ds, its reference ds', and threshold t.
Output: The number of correct C, incorrect I, extra E, and missing M elements.

```
for each element e1 in ds do
    for each element e2 in ds' do
        σ = calculateSimilarity(e1, e2);
        if σ == 1.0 then
            setUpCorrectAssignment(e1, e2); C++;
        else if σ > t then
            setUpIncorrectAssignment(e1, e2); I++;
        else
            E++;
        end
    end
end
for each element e2 in ds' do
    if e2 has no assignment then
        M++;
    end
end
```

attribute `first_name` actually contains a person's first name and no numeric value).

In QuaIIe, the formula by Logan et al. [39] 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:

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

B. Completeness

Completeness is broadly defined as the breadth, depth, and scope of information contained in the data [17] and can be divided into three subtypes. *Schema completeness* is the degree to which concepts and their attributes are present in a schema, *column completeness* defines the ratio of missing values to all values in a variable, and *population completeness* measures the missing values with respect to the real reference population [18]. A number of authors [15][18] calculate data completeness according to:

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

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 QuaIIe 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. [39]:

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

In addition, QuaIIe provides a variant of (4) that explicitly ignores duplicate entries:

$$Com(c, c') = \frac{|\text{unique}(c)|}{|\text{unique}(c')|}. \quad (6)$$

Oliveira et al. [43] provide the only formal definition of completeness (i.e., the missing value problem) for a relational schema $R(A)$ that contains a finite set of relations $r(A)$, where A is an attribute set $(a_1, a_2, ..., a_m)$, t a tuple, and $v(t, a)$ a specific value for tuple t and attribute a:

Definition 1: Let S be a set of attribute names, defined as: $S = \{a | a \in R(A) \wedge a$ is a mandatory attribute$\}$, i.e., $S \subseteq R(A)$. There is a missing value in attribute $a \in S$ iff: $\exists t \in r : v(t, a) = null$.

In contrast to other approaches, Definition 1 is restricted to `null` values and does not consider default or placeholder values (e.g., "NA" or "-99") as data incompleteness. Hinrichs proposed a metric, which corresponds to Def. 1 for different aggregation levels: attribute-value-level, record-level, concept-level, and DB-level. While the completeness of an attribute value $Com(v)$ is either 0 (if v=`null`) or 1 (else), completeness on record-level $Com(r)$ is the arithmetic mean of all attribute-value completeness measures for that record. The completeness of a concept (relation in [44]) is defined as

$$Com(c) = \frac{\sum_{i=1}^{n} Com(r_i)}{|c|}, \quad (7)$$

where n is the number of records in concept c. In addition, DB-level completeness is defined as the arithmetic mean of all concept-level completeness measures.

Schema completeness describes the extent to which real-world concepts of the application domain and their attributes and relationships are represented in the schema [18]. The metric for schema completeness in QuaIIe corresponds to the metric for data completeness in (5):

$$Com(s, s') = \frac{C_s + I_s}{C_s + I_s + M_s}. \quad (8)$$

Batista and Salgado [45] 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 [45] assume pre-defined schema mappings to be provided, QuaIIe calculates the distance or similarity values between the schema elements on-the-fly.

In addition, Nauman et al. [46] 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 [46]. In contrast, the schema completeness metric in QuaIIe 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 [40]) 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 [40] and in terms of correct, incorrect, extra, and missing records, is defined as:

$$Per(c, c') = \frac{C + I}{C + I + E}. \quad (9)$$

Schema pertinence describes a schema's relevance, which means that a schema with low pertinence has a high number of unnecessary elements [18]. 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 (9), schema pertinence is calculated in QuaIIe as

$$Per(s, s') = \frac{C_s + I_s}{C_s + I_s + E_s}, \tag{10}$$

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. Timeliness

An important aspect of many types of data is that the data values may change over time, for example, product pricing or customer addresses. In terms of time-related DQ dimensions, it can be distinguished between *currency* and *timeliness* [18]. According to [18], currency describes how promptly data is updated compared to changes in the real world. Ballou et al. [47] proposed the following metric for the currency of a single record r:

$$Cur(r) = (DeliveryTime(r) - InputTime(r)) + Age(r), \tag{11}$$

which requires meta information about each record. *DeliveryTime* is the time when the data is delivered to the customer, *InputTime* describes the time when the data is entered in the IS, and *Age* is the age of the data prior to system entrance. Since the age is rarely available and the delivery time would require in-depth domain knowledge, currency is calculated in QuaIIe as

$$Cur(c) = \frac{\sum_{r \in c} Now - InputTime(r)}{|c|}. \tag{12}$$

Here, *Age* is assumed to be 0 and the delivery time is assumed to be the timestamp of the data quality assessment. Additionally, the record-wise currency values are aggregated via average to assess the currency of a data set (i.e., concept). According to [18], timeliness describes how current the data is for a task at hand, which takes into account the specific use of a data value. An example would be the program of a cinema, which could be current because it contains the most recent data, but it is not timely if it is available after the start of the desired movie. Since QuaIIe focuses on the objectively measurable quality dimensions, timeliness is calculated according to:

$$Tim(c) = max\Big(0, 1 - \frac{Cur(c)}{Vol(c)}\Big), \tag{13}$$

omitting the aspect of the data usage. $Vol(c)$ is the *volatility* of a data set, which is a domain-specific value that describes how fast records become irrelevant [47]. Stock exchange prices (V3) that are updated by the second are considered highly volatile, while customer addresses display a low volatility as customers move every few years at maximum. Some types of data like birth dates have a volatility of 0, because they never become obsolete, therefore infinite timeliness is assumed in QuaIIe.

While schema evolution might cause a schema to become outdated, to the best of our knowledge, no approaches for measuring the outdatedness of schemas exist. Therefore, the current version of QuaIIe considers schemas to be time-invariant. It should be pointed out that this would be an interesting topic to be considered as future work.

E. 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 [18]. 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 [48]. 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 [48]: (1) *probabilistic assignment* using the Fellegi-Sunter model [49], (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 QuaIIe, 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 [50]. 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 $\delta = 1 - \sigma$. All distance and similarity values produced by QuaIIe can be assumed to be normalized over the unit interval of real numbers [0,1].

Since each data record consists of multiple attribute values, the distance function is a weighted-average of individual attribute distance functions. QuaIIe 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 adds 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. QuaIIe sets a predefined clustering threshold, which has been evaluated in experiments presented in [36]. 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, QuaIIe 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. QuaIIe 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 [51] for further information on hierarchical clustering.

Following, the minimality metric in QuaIIe 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

$$Min(c) = \begin{cases} 1.0, & \text{if } |c| = 1 \\ \frac{|clusters|-1}{|c|-1}, & \text{else} \end{cases}. \quad (14)$$

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. QuaIIe 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 [21] 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 [52], 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, (14) can be reformulated for schema minimality according to

$$Min(s) = \begin{cases} 1.0, & \text{if } |s| = 1 \\ \frac{|clusters|-1}{|s|-1}, & \text{else} \end{cases}, \quad (15)$$

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

F. 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 [41]. 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 QuaIIe, 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 [53].

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 [54], 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 [41]. 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.

G. Readability

The current version of QuaIIe supports readability on schema-level only. Although we did not find any discussion on readability on data-level, the current implementation of the readability could be used to evaluate the readability of string values on content-level likewise.

A schema should be readable, which means it should represent the modeled domain in a natural and clear way so it is self-explanatory to the user [41]. Good readability supports semantic expressiveness and enables automatic mappings to other domains that are based on dictionary approaches

(e.g., by using publicly available online dictionaries such as WordNet [55] or DBpedia [56]). While the readability of a conceptual schema in its graphical representation also includes aesthetic criteria, such as the arrangement of entities or crossing lines, the readability of a logical schema is limited to the actual naming of entities and relationships. Since clarity is subjective, no generally valid formal definition for this quality dimension exists [18].

We suggest two core measures to guarantee a sufficient level of readability. The fist measure is based on a validation of all schema element names using a dictionary approach, where we selected WordNet [55] for the implementation in QuaIIe. As a general rule, concepts should usually be described by singular nouns, while relationships should be described in present tense verbs or by a combination of two nouns [57]. A user should be permitted to add company-specific abbreviations that are widely used in practice and are usually not contained in public dictionaries. This is currently allowed in form of a CSV (comma-separated values) file. The second measure is a mandatory set of readability rules with which a rule checker can verify compliance. Both, readability rules and compliance with the online dictionary, can be formulated in criteria $crit$. Those criteria are applied to "words", which are extracted from schema element labels (e.g., an attribute name). A word can be either the complete name of an element, or part of it. If a schema uses delimiters like underscores (_) or hyphens (-), one string is split into several words. For example, `first_name` is split into "first" and "name". The following list contains a set of exemplary criteria, which can be used as a starting point and extended by additional domain-specific criteria.

- *Dictionary existence:* whether the word can actually be found in WordNet.
- *Consistent naming:* check if the naming style is consistent, e.g., only upper case, initial upper case, lower case, camel case, with or without blanks and/or hyphens.
- *Hypernyms:* if the word has hypernyms in the schema.
- *Synonyms:* if the word has synonyms in the schema.
- *Dates:* if the term "date" occurs in an attribute name, its data type must be `dateTime`.
- *Identifier:* if the term "ID" or "Identifier" occurs in an attribute name, the attribute must be a primary key or at least unique.
- *Foreign keys:* the naming of a foreign key and its corresponding primary key must be equivalent.

Based on these criteria, we suggest calculating a readability score according to

$$Red(s) = \frac{\sum_{i=1}^{|w|} \#\text{fcrit}_i / \#\text{crit}}{|w|}, \qquad (16)$$

where $|w|$ is the total number of words considered, $\#crit$ is the number of considered criteria, and $\#fcrit_i$ is the number of fulfilled criteria per word w_i.

H. 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 $\times$ [53]) 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 [37]. 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 II and III. 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

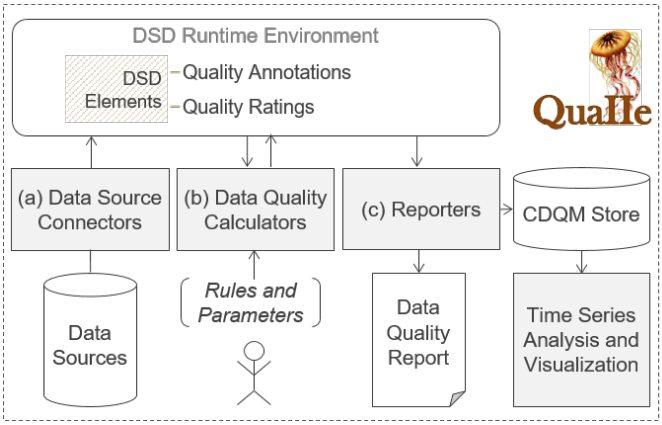

Fig. 1. Implementation Architecture of QuaIIe

Fig. 1 shows the architecture of our modular Java-based tool QuaIIe for measuring IIS data-level and schema-level quality ad hoc and continuously. The tool consists of three abstract components: (a) data source connectors to establish an IS connection and to load schema information in form of DSD elements, (b) quality calculators that perform schema and data

TABLE II
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 III
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	$c1 \times c_2$	$\frac{Com(c_1) * Com(c_2)}{1 + Com(c_1) * \left(\frac{Com(c_2)}{Acc(c_2)} - 1\right) + Com(c_2) * \left(\frac{Com(c_1)}{Acc(c_1)} - 1\right) + \left(\frac{Com(c_1)}{Acc(c_1)} - 1\right) * \left(\frac{Com(c_2)}{Acc(c_2)} - 1\right)}$

quality measurement and annotate this information to the DSD elements, and (c) reporters, which either generate a human- and machine-readable quality report, or persist the continuous DQ measurements in a DB for later analysis. 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, QuaIIe also allows user input in form of rules and parameters for specific quality calculations. The DSD runtime environment, which is described in more detail in Section IV-B, holds schema information on the loaded data sources in form of DSD elements along with assigned quality ratings and annotations. The DSD runtime environment exists only during the Java runtime, that is, during the execution of one (initial) DQ measurement procedure. In order to persist the DQ measurements for later analysis, they need to be exported to the CDQM store using one of the the reporters. In the following paragraphs, the selection of data sources used for our demonstration is justified and explained, and each component (a), (b), and (c), as well as the DSD runtime environment and the CDQM store are described in more detail.

A. Demonstration Data Sources

Recently, a call for more empiricism in DQ research has been made in [58], promoting (1) the evaluation on synthetic data sets to show the reproducibility of the measurements, and (2) evaluations on large real-world data sets. In order to cover both requirements, we used curated demo data sources of known quality, which allow manual verification of quality measurements, as well as real-world data sources of unknown quality. We selected those data sources also in a way to demonstrate the usage of all three volatility types (V1-3), from highly volatile, to completely static. All six used data sets are described in the following paragraphs in more detail.

a) Employees: The "employees" DB contains six tables with about three million records in total (300,024 in the table "employees" alone) and models the administrations of employees in a company [59]. Since it is a curated demo DB of known quality, we load the original employees DB into the `Datasource` object *dsEmpGS*, which is used as gold standard for our demonstration. In addition, we created two variants that have been automatically populated with randomly inserted errors in the original data: *dsEmp1* (501 records from the main table "employees") and *dsEmp2* (4,389 records). Table IV shows the error types that were used in the script. The added noise n is an absolute error that is normally distributed.

b) Sakila: The Sakila DB has 16 tables and models the administration of a film distribution [60]. While the employees

TABLE IV
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)}

DB contains a large number of records for quality measurement on the data-level, Sakila consists of a more advanced schema and is thus used for schema quality measurement that requires a GS (again, because of the known quality of the DB). We employed the `Datasource` object *dsSakilaGS*, which represents the original Sakila DB, as GS. In addition, we created three modified versions of the Sakila DB: one with minor modifications, but without removing or adding elements to affect correctness (*dsSakila1*), a second one where attributes and tables have been removed to affect completeness (*dsSakila2*), and a third one where attributes and tables have been added to affect pertinence (*dsSakila3*).

c) Northwind: The well-known Northwind DB from Microsoft (*dsNorthwind*) can be downloaded from [61] and demonstrates a comprehensive company DB, including 12 tables for e.g., customers, employees, products, order details. In addition to the original Northwind DB, we use an updated version published by dofactory [62] (*dsNorthwindNew*) with more recent dates, but only five tables.

d) CD CSV: We also used a CSV file (*dsCD*), which contains real-world music CD data that was originally published on the repeatability website by Felix Naumann [63]. The dataset contains information on the artist, title, genre, year, and contained tracks of 9,763 records, including 299 duplicates and was randomly extracted from freeDB.

e) Metadynea: To demonstrate data model independence of schemas investigated with QuaIIe, we also employed a Cassandra DB in productive use called "metadynea" (*dsMetadynea*). It contains five column families (tables) with about 60 GB of chemometrics data, which is distributed on three nodes.

f) Stock Exchange Data: In order to demonstrate the quality measurement of highly volatile data sets, we used real stock data (open, high, low, close, and volume) since the year 2000 from the equities of IBM, Microsoft, and Apple, accessed through the Alphavantage API (Application Programming Interface) [64]. The connected and analyzed data set is called *dsStockdata*.

As supplement to the demonstration in this paper, we published an executable (`QuaIIe.jar`) on our project website [65], 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 GS 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 DSD vocabulary. The transformation process from various data models and details of the DSD vocabulary are described in [21]. 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 runtime 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 type 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 of a 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. [66]). 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.

While this harmonization step enables comparability and quality measurement of schemas from different data models, it

```
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

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, five 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. [67]). 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 a subclass of `DSInstanceConnector`. Due to little meta data that is 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. [68]) 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.
- `ConnectorCassandra` uses the Datastax Java driver (cf. [69]) to access a Cassandra DB. This connector currently operates on schema-level only to evaluate the DQ measurements not only on relational data, but also on denormalized wide-column-store schemas.
- `ConnectorAlphavantage` is a domain-specific connector to crawl and load stock market data from the Alphavantage API [64].

Fig. 3 shows an example instantiation for each of the domain-independent 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 connCD = new ConnectorCSV("
    filepath/cd.csv", ",", "\n", "CD");
Datasource dsCD = connCD.loadSchema();

// Opening and loading a Cassandra data source
DSConnector connMeta = new ConnectorCassandra("
    hostname", "metadynea", "user", "pw");
Datasource dsMeta = connMeta.loadSchema();
```

Fig. 3. Data Source Connectors

In QuaIIe, each data source connector 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) `sakGS1`, a `DSDSchemaGS` that is solely used for schema quality measurement. Since specific gold standard implementations might have different tasks, each implementation requires a different set of parameters. However, all gold standards in QuaIIe inherit from the abstract class `GoldStandard`, 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 `DSDSchemaGS` 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.

C. Data Quality Calculators

Each DQ calculator is dedicated to one of the quality dimensions described in Section III and links the measurements to

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

// Creation of a schema gold standard
GoldStandard sakGS1 = new DSDSchemaGS(dsSakila1,
    dsSakilaGS);
```

Fig. 4. Gold Standards

the corresponding DSD elements in the DSD runtime environment. Quality measurements in the DSD runtime environment 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 for a DQ rating is 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 order to provide additional information about the quality. An example would be the annotation of functional dependencies to a concept. In the following, we summarize all DQ calculators that are currently implemented in QuaIIe and link them to the respective metrics from Section III, grouped by dimension:

- Accuracy / Correctness
  - `RefCorrectnessCalculator` (1) - data
  - `RatioAccuracyCalculator` (2) - data
  - `DSDCorrectnessCalculator` (3) - schema
- Completeness
  - `RatioCompletenessCalculator` (4) - data
  - `UniqueRatioCompletenessCalculator` (6) - data
  - `FilledCalculator` (7) - data
  - `DSDCompletenessCalculator` (8) - schema
- Pertinence
  - `RatioPertinenceCalculator` (9) - data
  - `DSDPertinenceCalculator` (10) - schema
- Timeliness
  - `AverageCurrencyCalculator` (12) - data
  - `AverageTimelinessCalculator` (13) - data
- Minimality / Duplicity
  - `RecordMinimalityCalculator` (14) - data
  - `SchemaMinimalityCalculator` (15) - schema
- Readability
  - `SchemaReadabilityCalculator` (16) - schema
- Normalform
  - `NormalFormCalculator` - schema

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 demonstration, 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 DSD runtime environment. One data quality rating or annotation is uniquely identifiable 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.

```
// Select concept "employees" from employees DB
Concept c = dsEmp1.getConcept("employees");

// Create a custom distance measure
EnsembleDistance<Record> dist = new EnsembleDistance
    <Record>();
dist.addDistance(new StringRecordDistance(c.
    getAttribute("first_name"), new
    LevenshteinDistance()), 0.5);
dist.addDistance(new StringRecordDistance(c.
    getAttribute("last_name"), new
    LevenshteinDistance()), 0.5);

// Perform quality calculations
RatioAccuracyCalculator.calculate(c,empGS,connEmp1);
RatioCompletenessCalculator.calculate(c,empGS,
    connEmp1);
RatioPertinenceCalculator.calculate(c, empGS,
    connEmp1);
RecordMinimalityCalculator.calculate(c, dist, 0.1,
    connEmp1);

// Retrieve DQ measurements from the DSD runtime
    environment (formerly ''Data Quality Store'')
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 V. The low quality values for accuracy and completeness result from the small subsets of the erroneous tables in contrast to the original employees table with 300,024 records.

TABLE V
DQ MEASUREMENT OF ERRONEOUS DATA SOURCES

Dimension	Metric	*dsEmp1*	*dsEmp2*
Accuracy	Ratio	0.0013	0.0116
Completeness	Ratio	0.0013	0.0116
Pertinence	Ratio	0.7725	0.7938
Minimality	Record	0.7180	0.7532

The time-related DQ dimensions currency and timeliness require information about the last update of a tuple, which is not available in the employees DB, but offered by Sakila DB in

form of an attribute `last_update`. We show the calculation in Fig. 6, which leads to a rating of 3.7156 for currency and 0.9988 for timeliness. Both calculators require information about the attribute that holds the update information. In addition, the timeliness calculation requires a parameter for the volatility. In the demonstration, the volatility is set to a value of 10 years for actor names, since it is very unusual that a name for an actor changes within that time frame.

We want to point out that due to their definition and in contrast to the other DQ dimensions, it is not possible to assess the time-related dimensions without prior knowledge, which does not align with the objective of QuaIIe to be domain-independent. However, we included those two DQ calculators in order to provide comprehensive DQ measurement and think it is worth investigating both dimensions in more detail in order to come up with automatically suggested parameters (e.g., for the volatility).

```
1 Concept actor = dsSakila1.getConcept("actor");
2 double volatility = 10 * MILLIS_IN_SEC * SEC_IN_MIN
      * MIN_IN_HOUR * HOUR_IN_DAY * DAY_IN_YEAR;
3
4 AverageCurrencyCalculator.calculate(connSakila1,
      actor, r -> (Date) r.getField("last_update"));
5 AverageTimelinessCalculator.calculate(connSakila1,
      actor, r -> (Date) r.getField("last_update"),
      volatility);
```

Fig. 6. Time-Related Data Quality Calculations

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 [65], 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 *DSD 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 explained in [36]. Fig. 7 shows the application of the schema quality calculators correctness, completeness, pertinence, minimality, and normalization.

```
1 DSDCorrectnessCalculator.calculate(dsSakila1, sakGS1
      );
2 DSDCompletenessCalculator.calculate(dsSakila2,
      sakGS2);
3 DSDPertinenceCalculator.calculate(dsSakila3,sakGS3);
4 RatioMinimalityCalculator.calculate(dsSakilaGS);
5 NormalFormCalculator.calculate(dsSakilaGS,
      connSakilaGS);
```

Fig. 7. Schema Quality Calculations

The results of the schema quality measurements applied to Sakila DB, employees DB, stock data, Northwind DB and Metadynea DB are provided in Table VI. The results are discussed in more detail in the following subsections.

TABLE VI
SCHEMA QUALITY MEASUREMENT RESULTS

Schema	Cor	Com	Pert
dsSakilaGS	1.0	1.0	1.0
dsSakila1	0.813	1.0	1.00
dsSakila2	0.929	0.813	0.929
dsSakila3	0.824	0.938	0.882

	Min	NF (t.=tables)
dsSakilaGS	1.0	6 t.: BCNF, 9 t.: 2NF, 1 t.: 1NF
dsEmpGS	0.8	All BCNF
dsStockdata	1.0	1 t.: BCNF
dsNorthwind	1.0	6 t.: BCNF, 4 t.: 2NF, 1 t.: 1NF
dsMetadynea	0.667	Not applicable for Cassandra

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 0.938 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 $\frac{13}{13+3+0} = 0.813$.

b) Schema Completeness: The completeness calculation was performed by removing schema elements. The DQ report for this demonstration 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 0.692 between `staff` and its correspondence in the GS. Consequently, both tables were not mapped because they were too different and completeness dropped to $\frac{13+0}{13+0+3} = 0.813$.

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 0.941. 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 $\frac{15+0}{15+0+2} = 0.882$. 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 0.833 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 (0.941) 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.

The results of all four schema quality measurement results are summarized in Table VI and elaborated in more detail in the following paragraphs.

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 [52] and DSD similarity, which we use in the following calculations with standard parameters that have been evaluated in [36]. The schemas of the Sakila DB (sakila_gs.ttl), the Northwind DB and stock data achieve an ideal minimality rating of 1.0, because all schema elements are sufficiently different to each other. For the Sakila DB with 16 schema elements, minimality is calculated according to $\frac{16-1}{16-1} = 1.0$.

However, the minimality ratings of the employees DB and metadynea are clearly below 1.0. This rating is expected for a Cassandra DB schema like metadynea, where denormalization (and thus, redundancy at the schema-level) is an intended design decision. In order to investigate the reason behind the minimality rating for the employees schema in more detail, we observe the similarity matrix from the DSD similarity in Table VII. 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. [59]). 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".

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 VIII. Due to the considerable 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 [54]. 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 to support full FD discovery (i.e., without approximation).

TABLE VII
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".

TABLE VIII
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. 8 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.

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. 9). Additionally, it is possible to estimate the quality (cf. Section III-H), which is not a complete measurement of the new integrated concept, but

```
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. 8. Data Integration

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.

```
DSInstanceConnector integrConn = new
    IntegratedInstanceConnector(ic);

RatioCompletenessCalculator.calculate(ic, gsEmp,
    integrConn);
RatioAccuracyCalculator.calculate(ic, gsEmp,
    integrConn);
RatioPertinenceCalculator.calculate(ic, gsEmp,
    integrConn);

RatioCompletenessCalculator.estimate(ic);
RatioAccuracyCalculator.estimate(ic);
RatioPertinenceCalculator.estimate(ic);
```

Fig. 9. DQ Estimation of an Integrated Concept

The ratings for the DQ calculations and estimations from Fig. 9 are compared in Table IX 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 IX
DQ CALCULATION OF AN INTEGRATED CONCEPT

Dimension	Metric	Measurement	Estimation
Accuracy	Ratio	0.0129	0.0130
Completeness	Ratio	0.0129	0.0128
Pertinence	Ratio	0.7916	0.7916

E. Data Quality Reports

In order to present the quality ratings and annotations contained in the DSD runtime environment 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. 10 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 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. [65]), the output is such a report.

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

Fig. 10. Data Quality Report Generation

The exemplary quality report in Fig. 11 shows an excerpt of the assessed data source *dsEmp1*, which illustrates possible quality ratings and annotations on schema (here: completeness, minimality, and normalization) and on the data-level (here: accuracy and pertinence).

```
<DataQualityReport>
 <Datasource label="employees" URI="http://example.
     com/dsEmp1">
  <Quality>
    <Ratings>
      <Completeness>
        <DSDSchema>1.0</DSDSchema>
      </Completeness>
      <Minimality>
        <Ratio>0.8</Ratio>
      </Minimality>
    </Ratings>
  </Quality>
  <Concept label="employees" URI="http://example.com
      /dsEmp1/employees">
    <Quality>
      <Ratings>
          <Accuracy>
           <Ratio>0.0013</Ratio>
        </Accuracy>
        <Pertinence>
           <Ratio>0.7725</Ratio>
        </Pertinence>
      </Ratings>
      <Annotations>
        <Candidate_Key>[{emp_no}]</Candidate_Key>
        <Normal_Form>BCNF</Normal_Form>
      </Annotations>
    </Quality>
    <Attribute label="emp_no" URI="http://example.
        com/dsEmp1/employees/emp_no">
    <Quality>
      <Ratings>
        <Key_Attribute>
          <Pseudo_Boolean>1.0</Pseudo_Boolean>
        </Key_Attribute>
      <Ratings>
    </Quality>
    </Attribute>
   ...
  </Concept>
  ...
 </Datasource>
 ...
</DataQualityReport>
```

Fig. 11. XML Data Quality Report

The first level below the root node lists all connected data sources that contain quality ratings for the entire `Datasource` as well as concepts and associations as child nodes. Concepts and associations are further subdivided into their comprising attributes and again, quality ratings and

annotations on concept- or association-level, respectively. Attributes constitute the deepest level in the schema-level hierarchy and can contain quality information on their own, e.g., whether an attribute is a key attribute or not.

F. Continuous Data Quality Measurement

In order to demonstrate the practical application of CDQM with QuaIIe, we assume the following use cases, ordered by data volatility type: (V1) monitoring the conformance of Wikipedia data to a static gold standard, (V2) measuring the quality of sales data that is loaded on a daily basis with an ETL job into a Data Warehouse, and (V3) continuous measurement of stock data to support data analytics. All three use cases are fictional stories, fueled by real-world data, in order to provide demonstrative examples how CDQM can be applied to different types of data. Fig. 12 depicts the Entity-Relationship diagram of our CDQM store, which follows the suggestion for such a store proposed in [22]. Each CDQM measurement is uniquely identified by the respective DSD element ID (which is an URI), the metric ID, and a timestamp when the measurement was taken. Thus, for a specific CDQM chart, it is only necessary to query for the desired element, metric, and time-range to plot the information for a user.

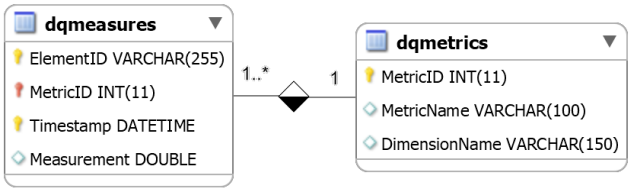

Fig. 12. ER Diagram of the CDQM Store

1) CDQM for Static Data: Assume a Wikipedia administrator wants to automatically monitor the state of the articles about Municipal districts in Ireland [70]. The list in [70] provides him with a gold standard in terms of completeness, since it contains all names of Municipal districts in Ireland. This data set is considered static (type V1), because the addition, removal, or merging of Municipal districts is an extremely rare event. In an idealized state (completeness= 1.0), there exists a Wikipedia article for each district. The administrator monitors the completeness of these articles between the years 2000 and 2010 and experiences a typical completeness development for static data, illustrated in Fig. 13. The completeness grows strictly monotonous, which is due to the fact that articles are never removed or outdated and the gold standard remains constant (no new Municipal districts arise). This use case also shows that CDQM can be applied to semi-structured data.

2) CDQM of Incoming Sales Data: In a large company, sales data (here a derivative of the Northwind DB [61]) is collected every batch-wise from different sales persons to be loaded into the central Data Warehouse (DWH). The DWH administrator wants to monitor the quality of each incoming file and the central DWH. Fig. 14 shows the currency of the

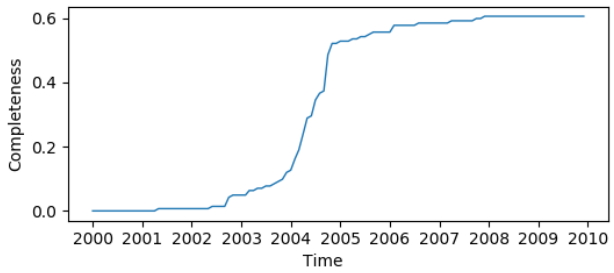

Fig. 13. Completeness of Data extracted from Wikipedia

DWH, which achieves constantly higher values, because the average age of a sale increases as time progresses and the DWH contains more and more older sales. However, intuitively the quality of the DWH is not deteriorating as the sales data does not get less trustworthy. This indicates that currency is probably a too naive measure in the context of accumulating data. In the lower half of Fig. 14, the completeness of the DWH and the average completeness of collected data per day are shown. While the variance of the completeness of the newly arriving data is consistently high, the variance of the completeness for the DWH decreases and almost converges around 0.95.

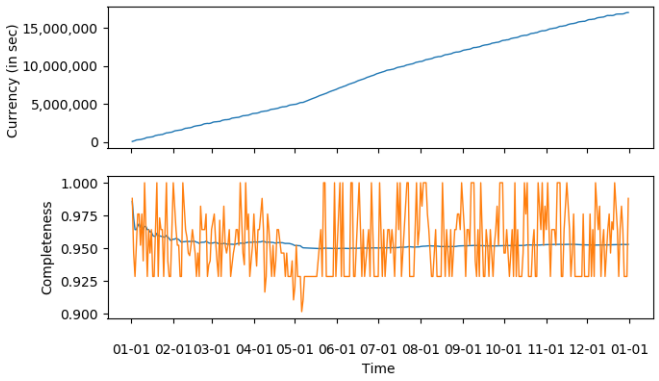

Fig. 14. Currency and Completeness of Sales Data

3) CDQM for Data Analytics: Assume, a stock market data analyst creates a machine learning (ML) model that uses the last 50 records of 5-minute-interval stock data, which is of volatility type V3. The model is always built 2 minutes after a new record arrives and predicts the further development for a specific equity (in our example we used IBM). In order to ensure the validity of the model result, we continuously measure the quality of the employed data, each time the model is created. Fig. 15 shows metrics for the DQ dimensions completeness and currency for 10 days between July 10th and 20th, 2018. While the completeness rating remains constantly at 1.0, i.e., all records are constantly delivered, the currency varies. This is an example for intended variation in the quality chart. Since currency describes how up-to-date the data is at hand, and the stock market is closed over the weekend (big

peak) and after 4 pm (small peaks), the model prediction with respect to currency is poor when the market closes and reaches a stable phase during the day. Since during one day only 78 5-minute records are delivered, the data analyst can create only about 28 models where the data has the best currency.

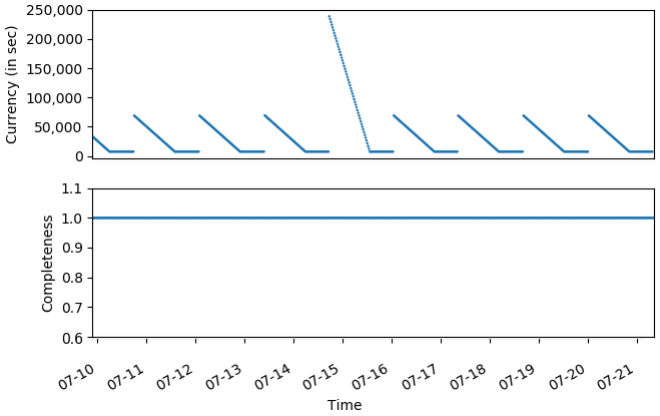

Fig. 15. Currency and Completeness of Stock Data for ML Model

V. Conclusion and Future Work

In this paper, we have demonstrated how to measure data quality (1) ad hoc and without preparation activities, and (2) continuously over time, using our previously developed DQ tool QuaIIe [1]. QuaIIe covers metrics for the DQ dimensions accuracy, correctness, completeness, pertinence, timeliness, minimality, readability, and normalization on both, data-level and schema-level of an IS. In addition, it is possible to estimate the quality of integrated IS. To the best of our knowledge, there exists no other DQ tool that implements such a large number of different dimensions and supports ad hoc as well as continuous DQ measurement in a single application. The major contribution in this paper is to highlight the importance for initial ad hoc DQ measurement to get a first insight into DQ, as well as continuous DQ measurement over time, which allows to observe how DQ evolves and to detect unexpected changes in the data.

As has been seen in Section IV-F, the time-progression of some DQ measures can diverge from an intuitive understanding of data quality, e.g., the currency of aggregated data. Consequently, a need for specialized metrics for different types of data in CDQM arises.

Our ongoing work includes long-term evaluations of QuaIIe's continuous measurement functionality. In addition, a connector for Oracle DBs and a graphical user interface to visualize the DQ measurements are currently under development. We also plan to implement connectors for other NoSQL DB types like document stores and graph DBs to further evaluate the comparability of DQ measurements on different schema models.

Acknowledgment

The research reported in this paper has been partly supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry for Digital and Economic Affairs, and the Province of Upper Austria in the frame of the COMET center SCCH. In addition, the authors would like to thank Gudrun Huszar for the implementation of the readability calculator and Julia Hilber for her research work on the Cassandra connector.

References

[1] L. Ehrlinger, B. Werth, and W. Wöß, "QuaIIe: A Data Quality Assessment Tool for Integrated Information Systems," in *Proceedings of the Tenth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2018)*. International Academy, Research, and Industry Association (IARIA), 2018, pp. 21–31.

[2] KPMG International, "Now or Never: 2016 Global CEO Outlook," 2016.

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

[4] T. C. Redman, "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: November 2018].

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

[6] E. E. Ruppert, R. D. Barnes, and R. S. Fox, *Invertebrate Zoology: A Functional Evolutionary Approach*, 7th ed. Cengage Learning, 2003.

[7] The Apache Software Foundation, "Apache Cassandra," Online, http://cassandra.apache.org [retrieved: November 2018].

[8] D. C. Baulcombe, S. Chapman, and S. Santa Cruz, "Jellyfish Green Fluorescent Protein as a Reporter for Virus Infections," *The Plant Journal*, vol. 7, no. 6, 1995, pp. 1045–1053.

[9] 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.

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

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

[12] 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.

[13] V. S. V. Pulla, C. Varol, and M. Al, *Open Source Data Quality Tools: Revisited*. Springer International Publishing, 2016, pp. 893–902.

[14] L. Sebastian-Coleman, *Measuring Data Quality for Ongoing Improvement: A Data Quality Assessment Framework*. Newnes, 2012.

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

[16] N. R. Chrisman, "The Role of Quality Information in the Long-Term Functioning of a Geographic Information System," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 21, no. 2, 1983, pp. 79–88.

[17] 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, Mar. 1996, pp. 5–33.

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

[19] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for Data Quality Assessment and Improvement," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, 2009, p. 16.

[20] A. Maydanchik, *Data Quality Assessment*. Bradley Beach, NJ, USA: Technics Publications, LLC, 2007.

[21] 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.

[22] L. Ehrlinger and W. Wöß, "Automated Data Quality Monitoring," in *Proceedings of the 22nd MIT International Conference on Information Quality (MIT ICIQ 2017)*, J. R. Talburt, Ed., UA Little Rock, Arkansas, USA, 2017, pp. 15.1–15.9.

[23] B. Heinrich, D. Hristova, M. Klier, A. Schiller, and M. Szubartowicz, "Requirements for Data Quality Metrics," *Journal of Data and Information Quality*, vol. 9, no. 2, Jan. 2018, pp. 12:1–12:32.

[24] Microsoft Inc., "Task Scheduler," 2018, https://docs.microsoft.com/en-us/windows/desktop/taskschd/task-scheduler-start-page [retrieved: November 2018].

[25] AQR Capital Management, LLC, Lambda Foundry, Inc. and PyData Development Team, "Python Data Analysis Library - pandas," 2018, https://pandas.pydata.org [retrieved: November 2018].

[26] J. Hunter, D. Dale, E. Firing, and M. Droettboom, "Matplotlib," 2018, https://matplotlib.org [retrieved: November 2018].

[27] J. Adelman, M. Baak, N. Boelaert, M. D'Onofrio, J. A. Frost, C. Guyot, M. Hauschild, A. Hoecker, K. J. C. Leney, E. Lytken, M. Martinez-Perez, J. Masik, A. M. Nairz, P. U. E. Onyisi, S. Roe, S. Schaetzel, and M. G. Wilson, "ATLAS Offline Data Quality Monitoring," *Journal of Physics: Conference Series*, vol. 219, no. 4, 2010, p. 042018.

[28] V. Raman and J. M. Hellerstein, "Potter's Wheel: An Interactive Data Cleaning System," in *Proceedings of the 27th VLDB Conference, Roma, Italy*, vol. 1, 2001, pp. 381–390.

[29] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive Visual Specification of Data Transformation Scripts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 3363–3372.

[30] A. Rolland, "MobyDQ," 2018, https://github.com/mobydq/mobydq [retrieved: November 2018].

[31] Apache Software Foundation, "Apache Griffin," 2018, https://griffin.incubator.apache.org [retrieved: November 2018].

[32] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data Quality Assessment," *Computing Surveys (CSUR)*, vol. 45, no. 4, Apr 2002, pp. 211–218.

[33] D. P. Ballou and H. L. Pazer, "Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems," *Management Science*, vol. 31, no. 2, 1985, pp. 150–162.

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

[35] Oxford University Press, "Definition of Gold Standard in English," Online, 2017, http://www.oxforddictionaries.com/definition/american_english/gold-standard [retrieved: November 2018].

[36] L. Ehrlinger, "Data Quality Assessment on Schema-Level for Integrated Information Systems," Master's thesis, Johannes Kepler University Linz, 2016.

[37] 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.

[38] 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 (MIT ICIQ 2016)*, 2016, pp. 16.1–16.13.

[39] 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.

[40] G. Salton and M. J. Mcgill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

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

[42] O. Herden, "Measuring Quality of Database Schema by Reviewing – Concept, Criteria and Tool," in *Proceedings of 5th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, 2001, pp. 59–70.

[43] P. Oliveira, F. Rodrigues, and P. R. Henriques, "A Formal Definition of Data Quality Problems," in *Proceedings of the 10th International Conference on Information Quality (MIT ICIQ 2005)*, 2005.

[44] H. Hinrichs, "Datenqualitätsmanagement in Data Warehouse-Systemen [Data Quality Management in Data Warehouse Systems]," Ph.D. thesis, Universitt Oldenbourg, 2002.

[45] 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.

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

[47] D. Ballou, R. Wang, H. Pazer, and G. K. Tayi, "Modeling Information Manufacturing Systems to Determine Information Product Quality," *Management Science*, vol. 44, no. 4, 1998, pp. 462–484.

[48] 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, 2007, pp. 1–16.

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

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

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

[52] 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.

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

[54] 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, 2012, pp. 251–264.

[55] C. Fellbaum, "WordNet and Wordnets," in *Encyclopedia of Language and Linguistics*, A. Barber, Ed. Elsevier, pp. 2–665, https://wordnet.princeton.edu [retrieved: November 2018].

[56] DBpedia Association, "DBpedia," 2018, http://wiki.dbpedia.org [retrieved: November 2018].

[57] S. Hoberman, *Data Model Scorecard*. Technics Publications, LLC, 2015.

[58] 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, 2018, pp. 35–43.

[59] Oracle Corporation, "Employees Sample Database," Online, https://dev.mysql.com/doc/employee/en [retrieved: November 2018].

[60] Oracle Corporation, "Sakila Sample Database," Online, https://dev.mysql.com/doc/ sakila/en [retrieved: November 2018].

[61] Microsoft Inc., "Northwind and pubs Sample Databases for SQL Server 2000," 2018, https://www.microsoft.com/en-us/download/details.aspx?id=23654 [retrieved: November 2018].

[62] dofactory, "SQL Tutorial Sample Database," 2018, https://www.dofactory.com/sql/sample-database [retrieved: November 2018].

[63] F. Naumann, "CD Datasets," 2018, https://hpi.de/naumann/projects/repeatability/datasets/cd-datasets.html [retrieved: November 2018].

[64] Alpha Vantage Inc., "ALPHA VANTAGE," 2018, https://www.alphavantage.co [retrieved: November 2018].

[65] L. Ehrlinger, "Data Quality Assessment for Heterogenous Information Systems," Online, http://dqm.faw.jku.at [retrieved: November 2018].

[66] W3C Working Group, "RDF 1.1 Turtle," Online, 2014, https://www.w3.org/TR/turtle [retrieved: November 2018].

[67] Oracle Corporation, "MySQL Connectors," Online, https://www.mysql.com/products/connector [retrieved: November 2018].

[68] The Apache Software Foundation, "Apache Jena," Online, https://jena.apache.org [retrieved: November 2018].

[69] DataStax, "Datastax Java Driver for Apache Cassandra," 2018, https://docs.datastax.com/en/developer/java-driver/3.0/ [retrieved: November 2018].

[70] Wikipedia, "Municipal District (Ireland)," 2018, https://en.wikipedia.org/wiki/Municipal_district_(Ireland) [retrieved: November 2018].