Data Quality is Context Dependent*

Leopoldo Bertossi[1]**, **Flavio Rizzolo**[2]***, and **Lei Jiang**[3]

[1] Carleton University, Ottawa, Canada. bertossi@scs.carleton.ca
[2] Carleton University, Ottawa, Canada. flavio@scs.carleton.ca
[3] University of Toronto, Toronto, Canada. leijiang@cs.toronto.edu

Abstract. We motivate, formalize and investigate the notions of data quality assessment and data quality query answering as context dependent activities. Contexts for the assessment and usage of a data source at hand are modeled as collections of external databases, that can be materialized or virtual, and mappings within the collections and with the data source at hand. In this way, the context becomes "the complement" of the data source wrt a data integration system. The proposed model allows for natural extensions, like considering data quality predicates, and even more expressive ontologies for data quality assessment.
Topics. Data quality and cleansing.

1 Introduction

The assessment of the quality of a data source is context dependent, i.e. the notions of "good" or "poor" data cannot be separated from the context in which the data is produced or used. For instance, the data about yearly sales of a product with seasonal variations might be considered quality data by a business analyst assessing the yearly revenue of a product. However, the same data may not be good enough for a warehouse manager who is trying to estimate the orders for next month.

In addition, data quality is related to the discrepancy between the actual stored values and the "real" values that were supposed or expected to be stored. For instance, if a temperature measurement is taken with a faulty thermometer, the stored value (the measurement) would differ from the right value (the actual temperature), which was the one supposed to be stored. This is an example of *semantically inaccurate data* [3].

Furthermore, another type of semantic discrepancy occurs when *senses or meanings* attributed by the different agents to the actual values in the database disagree [19], as shown in the Example 1. In this paper, we focus on data quality (DQ) problems caused by this type of semantic discrepancy.

Example 1. Tom is a patient in a hospital. Several times a day his temperature is measured and recorded by a nurse. His doctor, John, wants to see Tom's temperature around noon every day, to follow his evolution. The information

* Research funded by the NSERC Strategic Network on BI (BIN, ADC05)
** Faculty Fellow of the IBM CAS. Also affiliated to University of Concepción (Chile).
*** Also affiliated to University of Ottawa.

that John needs appears in the *TempNoon* relation of Table 1, which contains the temperatures between 11:30 and 12:30 per day for each of John's patients.

TempNoon

	Patient	**Value**	**Time**	**Date**
1	Tom Waits	38.5	11:45	Sep/5
2	Tom Waits	38.2	12:10	Sep/5
3	Tom Waits	38.1	11:50	Sep/6
4	Tom Waits	38.0	12:15	Sep/6
5	Tom Waits	37.9	12:15	Sep/7

Table 1.

TempNoon

	Patient	**Value**	**Time**	**Date**
1	Tom Waits	38.5	11:45	Sep/5
2	Tom Waits	38.0	12:15	Sep/6
3	Tom Waits	37.9	12:15	Sep/7

Table 2.

John has additional *quality* requirements for the temperature measurements of his patients: they need to be taken by a certified nurse with an oral thermometer. On Sep/5, unaware of the new requirements, Cathy takes Tom's temperature at 12:10 with a *tympanal* thermometer and records the result as the tuple number 2 in Table 1. Since the instrument used does not appear in the view that John has of the data, he interprets the 38.2^{o}C value as taken with an *oral* thermometer.

This is an example of a discrepancy between the semantics of the value as intended by the data producer (38.2^{o}C taken with a tympanal thermometer) and the semantics expected by the data consumer (38.2^{o}C taken with an oral thermometer). This tuple should not appear in a quality table, i.e., one that satisfies John's quality requirements, since such a table would contain only temperatures taken with an oral thermometer.

A similar problem appears in the third tuple in Table 1: It was taken by a new nurse, Helen, who is not yet certified and, thus, does not satisfy one of the doctor's requirements. This tuple should not appear in a quality table containing only temperatures taken by certified nurses.

Table 2 fixes the problems of Table 1 with respect to the doctor's specification: The problematic second and third tuples do not appear in it.

How can we say or believe that Table 2 does contain only quality data? Prima facie it does not look much different from Table 1. This positive assessment would be possible if we had a *contextual database* containing the additional information, e.g., Tables 3, 4 and 5.

S (shift)

	Date	**Shift**	**Nurse**
1	Sep/5	morning	Susan
2	Sep/5	afternoon	Cathy
3	Sep/5	night	Joan
4	Sep/6	morning	Helen
5	Sep/6	afternoon	Cathy
6	Sep/6	night	Cathy
7	Sep/7	morning	Susan
8	Sep/7	afternoon	Susan
9	Sep/7	night	Joan

Table 3.

C (certification)

	Name	**Year**
1	Ann	2003
2	Cathy	2009
3	Irene	2000
4	Nancy	1995
5	Susan	1996

Table 4.

T (type)

	Nurse	**Date**	**Type**
1	Susan	Sep/5	Oral
2	Cathy	Sep/5	Tymp
3	Joan	Sep/5	Tymp
4	Helen	Sep/6	Oral
5	Cathy	Sep/6	Oral
6	Susan	Sep/7	Oral
7	Joan	Sep/7	Oral

Table 5.

The first relation contains the name of the nurses in Tom Waits' ward and the shifts they work in by day. These are the nurses taking the measurements; since it is a small ward there is only one nurse per shift with that task. The second relation records the name of the certified nurses in the ward and the year they got the certification. The last relation contains the type of thermometer each nurse is using by day (e.g., oral or tympanal); each nurse takes all temperature measurements of the day using the same type of thermometer. This contextual information allows us to assess the quality of the data in Tables 1 and 2. ■

In this paper we take seriously the intuition and experience that data quality is context dependent. Our formalization of context is given as a system of integrated data and metadata of which the data source under quality assessment is a particular and special component. More precisely, the context for quality assessment of data in a certain instance D of schema $\mathcal{S}$ is given by an instance I of a possibly different schema $\mathcal{C}$, which could be an extension of $\mathcal{S}$. In order to assess the quality of D, it has to be "put in context", which is achieved by *mapping* D (and $\mathcal{S}$) into the contextual schema and data. Actually, $\mathcal{C}$ can be more complex that a single schema or instance, namely a collection of database schemas and instances interrelated by data- and schema mappings.

In our framework, a quality database instance D could be seen as a "footprint" of a the contextual, extended database I. The possibly extra information in I is what gives context to- and explains the data in D. The contextual schema and data is not used to enforce quality of a given instance. Instead, it is used to: (a) Assess the quality of the data in the instance at hand; (b) Characterize the quality answers to queries; and (c) Possibly obtain those quality answers to a user query. All this is achieved by comparing the given instance D with instance I, virtual or material, that can be defined for the contextual schema on the basis of D, external sources that contribute with data to the contextual schema, and possibly additional data at the contextual level, as shown in Example 1.

Instance I above could be replaced by a much richer contextual description, e.g. a full-fledged ontology. Along this line, but still in a classic database scenario, we might define some additional *quality predicates* on $\mathcal{C}$ [19]. They could be used to assess the quality of the data in D (and also the quality of query answers from D as we will explore later).

The following contributions can be found in this paper: (a) A model of context for data quality assessment. (b) Its application to clean or quality query answering. (b) Its application to data quality assessment via some natural measures that emerge from the model. (d) Some algorithms for the previously mentioned tasks in a few particular, but common and natural cases. (e) The creation of a framework that can be naturally extended in subsequent work to include more general contextual ontologies and externally defined quality predicates.

The rest of the paper is organized as follows. In Section 2, we present a general framework for contextual data quality and illustrate it with a running example. In Section 3, we consider two special cases of the general framework where we assume we have a contextual instance I that we can use for quality assessment and present an algorithm for quality query answering under this assumption. In

Section 4, we explore more complex cases, e.g., where such contextual instances do not exist. We discuss related work in Section 5; and conclude and point out to our ongoing and future work in Section 6.

2 A Framework for Contextual Data Quality

Consider a relational schema $\mathcal{S}$, with relational predicates $R, \ldots \in \mathcal{S}$. This schema determines a language $L(\mathcal{S})$ of first-order predicate logic. In this paper we consider only monotone queries and views, e.g. conjunctive queries and unions there of, which we will usually write in non-recursive Datalog with built-ins [1]. We also consider an instance D of $\mathcal{S}$, with extension $R(D)$ for R, etc. If database instances are conceived as finite sets of ground atoms, then, for each $R \in \mathcal{S}$, $R(D) \subseteq D$. Instances $D, R(D), \ldots$ are those under quality assessment wrt to a *contextual system*.

In its simplest form, a contextual system consists of a contextual relational schema $\mathcal{C}$ that may include a set $\mathcal{B}$ of built-in predicates. We may have or not an instance for $\mathcal{C}$. In a more complex scenario, the contextual system may consist of several contextual schemas $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and also a set of external schemas $\mathcal{E}$ that can be used by the contextual system for the assessment of an instance D of $\mathcal{S}$.

In this general framework, the participating schemas are related by *schema mappings*, like those found, for instance, in virtual data integration systems (VDISs) [22, 4] or data exchange [21], or even more complex logical relationships, like those common in peer data management systems [5, 6]. (Cf. [13] for an analysis of the connections between these three areas.) Schema mappings take the form of correspondences between two formulas, like queries or view definitions, each of them containing predicates from a single or several schemas. In particular, the data source under assessment D will be mappings into the contextual schema.

A common form of association, or mapping, is of the form $\forall \bar{x}(S(\bar{x}) \rightarrow \varphi_{\mathcal{G}}(\bar{x}))$, where S is a relational predicate of a data source and $\varphi_{\mathcal{G}}(\bar{x})$ is a conjunctive query over a global relational schema $\mathcal{G}$. These association can be found in *local-as-view* (LAV) VDISs with open (or sound) sources. Another common form of association is of the form $\forall \bar{x}(\psi_{\mathcal{S}}(\bar{x}) \rightarrow G(\bar{x}))$, found in *global-as-view* (GAV) VDISs with open sources, where $\psi_{\mathcal{R}}(\bar{x})$ is a conjunctive query over the union $\mathcal{R}$ of the relational schemas at the sources, and G is a global relational predicate. In *global-and-local-as-view* (GLAV) VDISs with open sources, we find associations between views (or queries), of the form $\forall \bar{x}(\psi_{\mathcal{R}}(\bar{x}) \rightarrow \varphi_{\mathcal{G}}(\bar{x}))$.

Figure 1 illustrates this general scenario. The relations R_i in D are under quality assessment, which is done via the contextual schema $\mathcal{C}$, which has relational predicates $C_1, \ldots, C_m$. There is also a set $\mathcal{P}$ of *contextual quality predicates* (CQPs) $P_1, \ldots, P_k$, which are defined as views over $\mathcal{C}$ plus possibly external sources $E_1, \ldots, E_j$. In some cases, the combination of schemas $\mathcal{C}, \mathcal{P}, \mathcal{E}$ can be seen as a single, extended contextual schema. In other cases, it may be useful to tell them apart. The external predicates E_i could have "nicknames" E_i' in $\mathcal{C}$, with simple view definitions as mappings, of the form $\forall \bar{x}(E'(\bar{x}) \equiv E(\bar{x}))$ (or, in Datalog notation, $E'(\bar{x}) \leftarrow E(\bar{x})$).

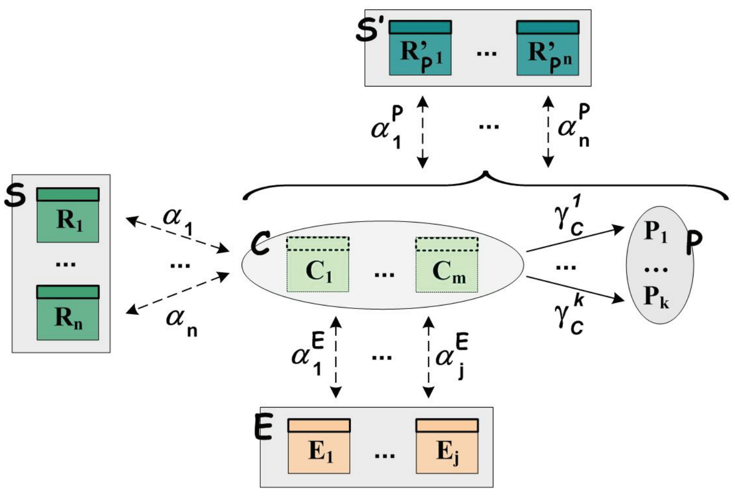

Fig. 1. General Framework for Contextual Data Quality

We will usually have a copy $\mathcal{S}'$ of schema $\mathcal{S}$, with relational predicates $R_1', \ldots, R_n'$. The idea is that the R_i' stand for the quality versions of the R_i, and their extensions can be compared. Those ideal predicates are related with the (extended) contextual schema by the α mappings. If the mappings involve quality predicates in $\mathcal{P}$, we usually use a predicate $R_{\mathcal{P}i}'$, and mappings $\alpha^{\mathcal{P}}$ to emphasize the dependency on $\mathcal{P}$. In principle, each of the mappings in Figure 1 can be of any of the LAV, GAV, GLAV kind, plus additional assumptions about openness/closedness of the involved instances, or a usual view definition.[4]

For example, the γ's are view definitions of the quality predicates in terms of the elements in $\mathcal{C}$, plus possible the external sources in $\mathcal{E}$. All these elements, as in the case of virtual data integration and peer data management systems, determines a collection of admissible instances I for the contextual instance $\mathcal{C}$.

In following sections we consider and study some relevant special cases of this general framework. In each of them, we address: (a) The problem of assessing the quality of the instance D consisting of the relations $R_1(D), \ldots, R_n(D)$. This has to do with analyzing how they differ from ideal, quality instances of the R_i. (b) The problem of characterizing and obtaining quality answers to queries that are expected to be answered by the instance D that is under assessment. As we will see, the two problems are related to each other. Actually, the former problem can be seen as a particular case of the latter.

We can present the main ideas already in the general framework: Given the different schemas involved, the mappings, and some materialized relations (e.g. the R_i, the E_k, some (parts) of the C_j) the relational predicates R_i' (or $R_{\mathcal{P}i}'$) will

[4] In virtual data integration it is possible to find semantics and algorithms for dealing with sources that coexist under different combinations of openness/closure assumptions [16, 4].

have (possibly virtual) admissible extensions, say $R_i'(I)$. The quality of $R_i(D)$ is determined by its "distance" to $R_i'(I)$, e.g. by the cardinality, $|R_i(D) \bigtriangleup R_i'(I)|$, of the symmetric difference. Different distance functions could be considered, specially if there are several admissible extensions for the R_i', as is often the case in VDISs, where several virtual *legal* instances may appear.

With respect to quality query answering, if a query $\mathcal{Q}$ is posed to D, but only quality answers are expected, the query could be rewritten in terms of the predicates R_i' and answered on the basis of their extensions.

Example 2. (example 1 continued) Consider query about patients and their temperatures around noon on Sep/5: $\mathcal{Q}(p, v)\ :\ \exists t \exists d(\mathit{TempNoon}(p, v, t, d) \land d = \text{Sep/5})$. The *quality answers* to this query posed to Table 1 should be $\langle$*Tom Waits*, 38.5$\rangle$, namely the projection on the first two attributes of tuple 1, but not of tuple 2 because it does not comply with the quality requirements according to the contextual tables 3, 4, and 5. Notice that if the same query is posed to Table 2 instead, which contains only quality data with respect to the quality requirements, we get exactly the same answer. ■

If there are several admissible instances, forming a class $\mathcal{I}$, the schema $\mathcal{S}'$ can be instantiated in them, obtaining instances $R'(I)$, with $I \in \mathcal{I}$. In consequence, $\mathcal{S}'(I) := \{R'(I) \mid R \in \mathcal{S}\}$ forms an instance for schema $\mathcal{S}'$. The *quality answers* to $\mathcal{Q} \in L(\mathcal{S})$, can thus be defined as those that are *certain*:

$$QAns_D^{\mathcal{C}}(\mathcal{Q}) = \{\bar{t} \mid \mathcal{S}'(I) \models \mathcal{Q}'[\bar{t}],\ \text{for all } I \in \mathcal{I}\} \tag{1}$$

where $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by replacing the R_i predicates by their copies, R_i'.

The notion of quality answer could be used to define the quality of instance D: For each of the relations $R \in \mathcal{S}$, we can pose the query $R(\bar{x})$ and obtain the quality answers $QAns(\mathcal{R})$. Each of the $QAns(\mathcal{R})$ becomes an instance for predicate R and can be compared with $R(D)$.

3 Instances as Views and Contextual Instances

As a broad and common case of the general framework, in addition to schema $\mathcal{S}$, the contextual schema $\mathcal{C}$, and an instance D of $\mathcal{S}$, we have the following:
(a) Each CQP $P \in \mathcal{P}$ defined as a conjunctive view, $P(\bar{x}) \leftarrow \gamma_{\mathcal{C}}(\bar{x})$, in terms of elements of $\mathcal{C}$ (and possibly built-in predicates). We denote with $\mathcal{C}^{\mathcal{P}}$ the schema $\mathcal{C}$ expanded with schema $\mathcal{P}$.
(b) For each database predicate $R \in \mathcal{S}$, a copy of it, R', which is defined as a conjunctive view of schema $\mathcal{C}^{\mathcal{P}}$:

$$R'_{\mathcal{P}}(\bar{x}) \ \longleftarrow \ \varphi_{R}^{\mathcal{C}}(\bar{x}),\ \varphi_{R}^{\mathcal{P}}(\bar{x}), \tag{2}$$

where $\varphi_{R}^{\mathcal{C}}(\bar{x}), \varphi_{R}^{\mathcal{P}}(\bar{x})$ are in their turn conjunctions of atomic formulas with predicates in $\mathcal{C}$, $\mathcal{P}$, respectively. A particular case is obtained when in (2) there are no CQPs in the view definition:

$$R'(\bar{x}) \ \longleftarrow \ \psi_{R}^{\mathcal{C}}(\bar{x}). \tag{3}$$

If we have an instance I for schema $\mathcal{C}$, then we will obtain a computed extensions $R'(I)$ and $R'_{\mathcal{P}}(I)$ by applying definitions (2) or (3). Now, if we also have an instance D of $\mathcal{S}$, the one under quality assessment, then $R(D)$ can be compared with $R'(I)$ and $R'_{\mathcal{P}}(I)$.

Intuitively, each CQP can be used to express an atomic quality requirement requested by a data consumer or met by a data producer. With the CQPs we can restrict the admissible values for certain attributes of tuples in I, so that only quality tuples find their way into D.

Although CQPs can be eliminated by unfolding their Datalog definitions, we make them explicit here, for several reasons: (a) To emphasize their role as predicates capturing quality requirements. (b) They allow us to compare data quality requirements in a more concrete way. For example, it is obvious that the quality requirement "temperature values need to be measured by an oral *or* tympanal thermometer" is less restrictive than "temperature values need to be measured by an oral thermometer". (c) Our approach allows for the consideration of CQPs that are not defined only in terms of $\mathcal{C}$ alone, but also in terms of other external sources, as indicated in Figure 1, that is, by view definitions of the form $P(\bar{x}) \leftarrow \gamma_{\mathcal{C}}(\bar{x}), \gamma_{\mathcal{E}}(\bar{x})$.

3.1 The simple case

A simple, restricted case of the general framework, and of the one in the previous section, in particular, occurs when the instance at hand D under assessment is exactly a materialized view of a contextual instance via a definition of the form (3). That is, for each $R \in \mathcal{S}$, we assume that $R(D) = R'(I)$. However, we may add additional quality requirements, thus obtaining an instance $R'_{\mathcal{P}}(I)$ via a view definition of the form (2). This would be an ideal instance of predicate R obtained from I using additional quality conditions.

In this case, $R(D) = R'(I)$, and $D(I) := \{R'(I) \mid R \in \mathcal{S}\} = D$. We also have the following instance for schema $\mathcal{S}$:

$$D_{\mathcal{P}}(I) = \{R'_{\mathcal{P}}(I) \mid R \in \mathcal{S} \text{ and } R'_{\mathcal{P}} \text{ is defined by (2)}\}. \qquad (4)$$

As expected, there may be differences between D and $D_{\mathcal{P}}(I)$. The latter is intended to be the clean version of D. Actually, it holds $R'_{\mathcal{P}}(I) \subseteq R'(I) = R(D)$, for each $R \in \mathcal{S}$.

Example 3. (example 1 continued) Schema $\mathcal{S}$ contains the database predicate *TempNoon(Patient, Value, Time, Date)* with the instance in Table 1 under assessment. The contextual schema $\mathcal{C}$ contains the database predicates S(*Date, Shift, Nurse*), T(*Nurse, Date, Type*) and C(*Name, Year*) introduced before. We have instances for them: Tables 3, 4 and 5, respectively. In addition, $\mathcal{C}$ contains predicate M(*Patient, Value, Time, Date, Instr*), which records the values of all measurements performed on patients by nurses (e.g., temperature, blood pressure, etc.), together with their time, date, instrument used (e.g., thermometer, blood pressure monitor), and the instance for it in Table 6.

M

	Patient	Value	Time	Date	Instr
1	T. Waits	37.8	11:00	Sep/5	Therm.
2	T. Waits	38.5	11:45	Sep/5	Therm.
3	T. Waits	38.2	12:10	Sep/5	Therm.
	...	...	...	...	...
4	T. Waits	110/70	11:00	Sep/6	BPM
5	T. Waits	38.1	11:50	Sep/6	Therm.

M (cont.)

	Patient	Value	Time	Date	Instr
6	T. Waits	38.0	12:15	Sep/6	Therm.
	...	...	...	...	...
7	T. Waits	37.6	10:50	Sep/7	Therm.
8	T. Waits	120/70	11:30	Sep/7	BPM
9	T. Waits	37.9	12:15	Sep/7	Therm.

Table 6.

Relation *TempNoon*(*Patient, Value, Time, Date*) can be seen as a materialized view over the instance in Table 6. It contains, for each patient and day, only temperature measurements close to noon.

According to (3), we can have the following view definition that captures the temperatures taken between 11:30 and 12:30:

$$\mathit{TempNoon}'(p, v, t, d) \leftarrow M(p, v, t, d, i), 11{:}30 \leq t \leq 12{:}30, i = \text{therm.} \qquad (5)$$

By materializing this view we obtain the instance shown in Table 1.

In order to express quality concerns, we now introduce some CQPs. In this way we will be in position to define the relation that contains only tuples satisfying the doctor's requirements, i.e., that the temperature has to be taken by a certified nurse using an oral thermometer. Accordingly, $\mathcal{P} = \{\ \mathit{Oral}(\mathit{Instr}), \mathit{Certified}(\mathit{Patient}, \mathit{Date}, \mathit{Time}),\ \mathit{Valid}(\mathit{Value})\}$.

In order to facilitate the definitions, we first introduce an auxiliary predicate, $\mathit{Temp}(\mathit{Patient}, \mathit{Date}, \mathit{Time}, \mathit{Nurse}, \mathit{Instr}, \mathit{Type})$, that compiles information about the temperature measurements, the instruments used, and the name of the nurses for the morning and afternoon shifts – we do not care at this point for the evening shift because it does not overlap with the 11:30-12:30 interval of interest. *Temp* associates to each measurement in M the nurse and type of thermometer used depending on the time at which the temperature was taken.

$$\begin{aligned}\mathit{Temp}(p, d, t, n, i, tp) \leftarrow\ & M(p, v, t, d, i), S(d, s, n),\ T(n, d, tp), i = \text{therm},\\ & 4{:}00 < t \leq 12{:}00, s = \text{morning}.\\ \mathit{Temp}(p, d, t, n, i, tp) \leftarrow\ & M(p, v, t, d, i), S(d, s, n),\ T(n, d, tp), i = \text{therm},\\ & 12{:}00 < t \leq 20{:}00, s = \text{afternoon}.\end{aligned}$$

With the help of this auxiliary predicate, the first two CQPs are defined by:

$$\mathit{Oral}(i) \leftarrow \mathit{Temp}(p, d, t, n, i, tp), tp = \text{oral}. \qquad (6)$$

$$\mathit{Certified}(p, d, t) \leftarrow \mathit{Temp}(p, d, t, n, i, tp),\ C(n, y). \qquad (7)$$

The first quality predicate is satisfied only when the instrument used is an oral thermometer. (The only instruments that appear in *Temp*'s tuples are thermometers and the additional requirement is specified by tp = oral). The second predicate can be used to specify that a measurement (uniquely identified by the patient, the date and the time) is made by a certified nurse.

A third CQP takes care of potential typing errors by checking that the temperature is in a predefined valid range. It is defined by:

$$\mathit{Valid}(v) \leftarrow M(p, v, t, d, i), 36 \leq v \leq 42. \tag{8}$$

With these three CQPs, we can define, according to (2), a new relation:

$$\begin{aligned} \mathit{TempNoon}'_{\mathcal{P}}(p, v, t, d) \leftarrow M(p, v, t, d, i),\ & 11{:}30 \leq t \leq 12{:}30, \\ & \mathit{Valid}(v), \mathit{Oral}(i), \mathit{Certified}(p, d, t). \end{aligned} \tag{9}$$

The new extension, for predicate $\mathit{TempNoon}'_{\mathcal{P}}$, is intended to contain only measurements satisfying the doctor's requirements, which corresponds to the instance shown in Table 2. ■

3.2 Quality query answering

Queries are written in the language associated to schema $\mathcal{S}$ and posed to instance D. However, clean answers to queries over D should be, in essence, the answers to the same query posed to $D'(I)$ or $D'_{\mathcal{P}}(I)$ instead. In consequence, and as a particular case of (1), for a query $\mathcal{Q}(\bar{x}) \in L(\mathcal{S})$, the set of *quality answers to $\mathcal{Q}$ wrt D* becomes:

$$QAns^{\mathcal{C}}_{D}(\mathcal{Q}) := \mathcal{Q}(D_{\mathcal{P}}(I)). \tag{10}$$

For monotone queries, e.g. conjunctive queries, it holds $QAns^{\mathcal{C}}_{D}(\mathcal{Q}) \subseteq Q(D)$, where the latter denotes the set of answer to Q from D.

Since the $R(D)$s are obtained as materialized Datalog views of the contextual instance I, query answering can be done via view unfolding. That is, in order to get quality answers, we evaluate the original query on the clean relations $R'(I)$ via view unfolding:

Quality Unfold Algorithm: (QUA)

1. Replace each predicate R in $\mathcal{Q}$ by its corresponding R' (or $R'_{\mathcal{P}}$), obtaining query $\mathcal{Q}'$.
2. Replace $\mathcal{Q}'$ by a query $\mathcal{Q}^{\mathcal{C}}_{\mathcal{P}} \in L(\mathcal{C} \cup \mathcal{P})$ via view unfolding based on (2).
3. If desired, or possible, unfold the definitions of the CQPs, obtaining the "quality query" $\mathcal{Q}^{\mathcal{C}} \in L(\mathcal{C})$, which can be evaluated on I.

The last step of the algorithm opens the possibility of considering CQPs that are not defined on top of schema $\mathcal{C}$ only. This is the case, for example, when they appeal to external sources, and also when they represent other kinds of *quality predicates*, e.g. of the form introduced in [19].

The quality of D could be assessed by comparing each $R(D)$ with the corresponding $R'_{\mathcal{P}}(I)$. This is just a particular case of quality query answering: The set of clean answers to each of the atomic queries $\mathcal{Q}(\bar{x}) : R(\bar{x})$ can be compared with the corresponding $R(D)$s.

Example 4. Consider the query on schema $\mathcal{S}$ of our running example that asks for the temperature of the patients on Sep/5: $\mathcal{Q}(p,v):\ \exists t \exists d(\mathit{TempNoon}(p,v,t,d) \wedge d = \text{Sep/5})$, which in Datalog notation becomes:

$$\mathcal{Q}(p,v) \leftarrow \mathit{TempNoon}(p,v,t,d),\ d = \text{Sep/5}. \tag{11}$$

To get quality answers, $\mathcal{Q}$ is rewritten in terms of schema $\mathcal{S}'$. Hence, $\mathcal{Q}$ is trivially rewritten as: $\mathcal{Q}'(p,v) \leftarrow \mathit{TempNoon}'_{\mathcal{P}}(p,v,t,d),\ d = \text{Sep/5}$. Now, $\mathit{TempNoon}'_{\mathcal{P}}$ is defined by (9). Thus, by view unfolding we get:

$$\begin{aligned}\mathcal{Q}^{\mathcal{C}}_{\mathcal{P}}(p,v) \leftarrow M(p,v,t,d,i),\ 11{:}30 \leq t \leq 12{:}30,\ d = \text{Sep/5},\\ \mathit{Valid}(v),\, \mathit{Oral}(i),\, \mathit{Certified}(p,d,t).\end{aligned} \tag{12}$$

This query can be evaluated directly on I, which contains relation M, by unfolding the definitions (6)-(8) of the quality predicates or directly using their extensions if they have been materialized. ■

Notice that in (12) we could have quality predicates that are not defined only in terms of the contextual schema $\mathcal{C}$, but defined also in terms of other external sources. In consequence, the query cannot be evaluated on I alone, and this might trigger requests for additional data.

Example 5. (example 4 continued) Consider now that, instead of having an instance for $C(\mathit{Nurse}, \mathit{Year})$, we have its definition in terms of an external source $\#X(\mathit{Nurse})$ that contains information about certified nurses; $\#X(\mathit{Nurse})$ returns *true* if the input nurse appears in the source, and *false* otherwise. Since $C(\mathit{Nurse}, \mathit{Year})$ is part of the definition of the *Certified* CQP (rule (7)), the evaluation of the unfolded query in (12) will trigger a request for data from $\#X$ in order to evaluate the *Certified* predicate. ■

This independence of the quality predicates from the contextual data or schema is particularly interesting in the case we want to use them to filter tuples from a relation, say R, in D. This situation can be easily accommodated in our framework, as follows. For predicate $R \in \mathcal{S}$, we consider a copy, or *nickname*, $R' \in \mathcal{C}$. Each R' shares the arity, the attributes of R, and their domains. We also have a simple GAV definition for R': $R'(\bar{x}) \leftarrow R(\bar{x})$, considering R as an *exact source*, in the terminology of virtual data integration [22] (this is usual in view definitions over a single instance). This creates a copy $R'(D)$ of $R(D)$ as a part of the contextual instance, and the contextual instance becomes $I := \{R'(D) \mid R \in \mathcal{S}\}$.

Now, if we want to obtain quality answers to a query $\mathcal{Q}$ on instance D (with schema $\mathcal{S}$), taking into account the quality predicates, we replace each predicate $R \in \mathcal{S}$ in $\mathcal{Q}$ by the conjunction $R'(\bar{x}) \wedge \varphi^{\mathcal{P}}_{R}(\bar{x})$. The data to evaluate the additional, quality formula $\varphi^{\mathcal{P}}_{R}(\bar{x})$ would be obtained from the external sources. The resulting query would be evaluated on I and extensions for the quality predicates. If the latter is missing, the query evaluation process could trigger *ad hoc* requests for external data.

In this section we considered the convenient, but not necessarily frequent, case where the instance D under assessment is a collection of exact materialized views of a contextual instance I. Alternative and natural cases we have to consider may have only a partial contextual instance I^- together with its metadata for contextual reference. We examine this case in Section 4.

4 Missing Contextual Data

Against what may be suggested by the examples above, we cannot assume that we always have a contextual instance I for schema $\mathcal{C}$. There may be *some* data for $\mathcal{C}$, most likely an incomplete (possibly empty) instance I^-, also data from other external sources, and the data in the instance D under assessment mapped into $\mathcal{C}$. In this more general case, a situation similar to those investigated in virtual data integration systems naturally emerges. Here, the contextual schema acts as the mediated, global schema, and instance D as a materialized data source. In the following we will explore this connection.

Let us now assume that we do not have a contextual instance I for schema $\mathcal{C}$, i.e. $I^- = \emptyset$. We could see D as a data source for a virtual data integration system, $\mathfrak{C}$, with a global schema that extends the contextual schema $\mathcal{C}$ [22, 4]. We may assume that all the data in D is related to $\mathfrak{C}$ via $\mathcal{C}$, but $\mathfrak{C}$ may have potentially more data than the one contributed by D and of the same kind as the one in D. In consequence, we assume D to be an *open source* for $\mathfrak{C}$. This assumption will be captured below through the set of intended or *legal* global instances for $\mathfrak{C}$.

Since not all the data in D may be up to the quality expectations according to $\mathcal{C}$, we need to give an account of the relationship between D and its expected quality version. For this purpose, as in the previous cases, we extend $\mathcal{C}$ with a copy $\mathcal{S}'$ of schema $\mathcal{S}$: $\mathcal{S}' = \{R' \mid R \in \mathcal{S}\}$. Now, $\mathcal{C}' := \mathcal{C} \cup \mathcal{S}'$, and it also becomes part of the global schema for $\mathfrak{C}$.

Definition 1. Assume each $R \in \mathcal{S}$ is defined as a Datalog view: $R(\bar{x}) \leftarrow \varphi^{\mathcal{C}}_{R}(\bar{x})$.
(a) A *legal instance* for system $\mathfrak{C}$ is an instance I' of the global schema, such that:
(a1) For every $R \in \mathcal{S}$, $R(D) \subseteq R(I')$; (a2) $I' \models \forall \bar{x}(R'(\bar{x}) \equiv \varphi^{\mathcal{C}}_{R}(\bar{x}) \wedge \varphi^{\mathcal{P}}_{R}(\bar{x}))$.
(b) An instance I of $\mathcal{C}$ is *legal contextual instance* (LCI) if there is a legal instance I' for $\mathfrak{C}$, such that $I = I'{\downarrow}\mathcal{C}$ (the restriction of I' to schema $\mathcal{C}$). ■

The condition in (a1) essentially lifts D's data upwards to $\mathfrak{C}$. The legal instances have extensions that extend the data in D when the views defining the Rs are computed. The sentences in (a2) act as global integrity constraints, i.e. on schema $\mathcal{C}'$, and have the effect of cleaning the data (virtually) uploaded to $\mathfrak{C}$.

We can also consider a variation of this case, where, in addition to D, we have only a fragment I^- of the potential contextual instances I. That is, we have a *incomplete* contextual data. In this case, Definition 1 has to be modified by adding the condition on I: (a3) $I^- \subseteq I'{\downarrow}\mathcal{C}$, which requires that the legal instance I' is "compatible" with the partial instance I^- at hand. With $I^- = \emptyset$, we obtain the previous case.[5]

[5] The occurrence of partial and materialized global instance I^- can be accommodated in the scenario of VDISs by considering I^- as a separate exact "source" for $\mathfrak{C}$.

Now, the idea is to pose queries in terms of the R', to obtain quality answers.

Definition 2. A ground tuple $\bar{t}$ is a *quality answer* to query $\mathcal{Q}(\bar{x}) \in L(\mathcal{S})$ iff $\bar{t} \in \bigcap\{\mathcal{Q}'(I) \mid I \text{ is an LCI}\}$, where $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by replacing every $R \in \mathcal{S}$ in it by R'. ■

As before, we denote with $QAns_D^{\mathcal{C}}(\mathcal{Q})$ the set of quality answers to $\mathcal{Q}$ from D wrt $\mathcal{C}$.

Example 6. Let us revisit the query $\mathcal{Q}(p, v)$ in (12) in Example 4. Let $\mathcal{Q}'(p, v)$ denote the same query, now expressed in terms of schema $\mathcal{S}'$. The instance of *TempNoon(Patient, Value, Time, Date)* in Table 1 is D, the instance under quality assessment.

We now define a VDIS $\mathfrak{C}$ with D as an open source, and the relations in Tables 3, 4 and 5 as forming a partial global instance I^-. In this case, there is no instance (relation) for predicate M(*Patient, Value, Time, Date, Instr*) in $\mathcal{C}$.

According to Definition 2, a quality answer to $\mathcal{Q}(p, v)$ has to be obtained *from every* LCI for $\mathfrak{C}$. Now, every LCI will contain tuples from *TempNoon(Patient, Value, Time, Date)* satisfying the conditions imposed by (9). In fact, Table 2 corresponds to the smallest LCI for $\mathfrak{C}$: No subset of it is an LCI and any superset satisfying (9) is also an LCI. In consequence, the first tuple in Table 2 is the only one satisfying the additional query condition $d = \text{Sep/5}$. We obtain: $QAns_D^{\mathcal{C}}(\mathcal{Q}(p, v)) = \{\langle \text{Tom Waits}, 38.5\rangle\}$. ■

Since we have the original instance D defined as an open source, we can take advantage of any of the existing algorithms for the computation of the certain answers to global queries under the openness assumption [17]. Since we are assuming that queries and view definitions are conjunctive, we can use, e.g. the *inverse rules algorithm* [14] or extensions thereof [4, 11]. We illustrate its application with an example.

Example 7. (example 6 continued) If we invert the definition of *TempNoon* in (5), we get:

$$M(p, v, t, d, i) \leftarrow \mathit{TempNoon}(p, v, t, d), 11{:}30 \leq t \leq 12{:}30, i = \text{therm}. \qquad (13)$$

We can evaluate $\mathcal{Q}_{\mathcal{P}}^{\mathcal{C}}(p, v)$ in (12) by unfolding the definition of predicate M according to (13), obtaining:

$$\begin{aligned}\mathcal{Q}_{\mathcal{P}}^{\mathcal{C}}(p, v) \leftarrow\ & \mathit{TempNoon}(p, v, t, d),\ 11{:}30 \leq t \leq 12{:}30,\ i = \text{therm},\ d = \text{Sep/5},\\ & \mathit{Valid}(v), \mathit{Oral}(i), \mathit{Certified}(p, d, t).\end{aligned}$$

The rewritten query can be now evaluated on the instances of *TempNoon*, S, C, and T (Tables 1, 3, 4 and 5, respectively). This produces the same result as in the previous example. ■

Finally, wrt data quality assessment, some alternatives naturally offer themselves. If we want to assess D, we can consider, for each LCI I, the instance

$\mathcal{S}'(I) := \{R'(I) \mid R \in \mathcal{S}\}$. We have $\mathcal{S}'(I) \subseteq D$. A possible quality measure could be $QM_1(D) := (|D| - max\{|\mathcal{S}'(I)| : I \text{ is LCI}\})/|D|$, inspired by the G_3 measure in [20].

Another possible measure is based, as suggested above, on quality query answering: For each predicate $R \in \mathcal{S}$, compute the query $\mathcal{Q}_R : R(\bar{x})$. Then, compute $QM_2(D) := (|D \smallsetminus \bigcup_{R \in \mathcal{S}} QAns_D^{\mathcal{C}}(\mathcal{Q}_R)|)/|D|$. The analysis and comparison of these and other possible quality measures are left for future work.

5 Related Work

The study on data quality spans from the characterization of types of errors in data (e.g., [25]), to the modeling of processes in which these errors may be introduced (e.g., [2]), to the development of techniques for error detection and repairing (e.g., [7]). Most of these approaches, however, are based on the implicit assumption that data errors occur exclusively at the syntactic/symbolic level, i.e., as discrepancies between data values (e.g., Kelvin vs. Kelvn).

As argued in [19], data quality problems may also occur at the semantic level, i.e., as discrepancies between the meanings attached to these data values. More specifically, according to [19], a data quality problem may arise when there is a mismatch between the *intended* meaning (according to its producer) and interpreted meaning (according to its consumer) of a data value. A mismatch is often caused by ambiguous communication between the data producer and consumer; such ambiguity is inevitable if some sources of variability (e.g., the type of thermometer used and the conditions of a patient) are not explicitly captured in the data (or metadata). Of course, whether or not such ambiguity is considered a data quality problem depends on the purpose for which the data is used.

In [19] a framework was proposed for *defining* both syntactic- and semantic-level data quality in an uniformed way, based on the fundamental notion of signs (values) and senses (meanings). A number of "macro-level" quality predicates were also introduced, based on the comparison of symbols and their senses (exact match, partial match and mismatch). In this work, we take the next step to propose a specific mechanism for capturing and comparing semantic-level data quality requirements using context relations and quality predicates, and show how they are *used* in query answering.

Use of contexts in data management has been proposed before (see [9] for a survey). Of course, there are different ways to capture, represent and use contexts. For instance, contextual information has been used to support a semi-automatic process for view design (see [8] for an overview). A context in [8] consists of a number of context elements, which are essentially attribute-value pairs (e.g., role='agent', situation='on site', time='today'); certain constraints can also be specified on a context (e.g., when role is 'manager', situation cannot be 'on site').

A context specification allows one to select from a potentially large database schema a small potion (a view) that is deemed relevant in that context. Given a context specification, the design of a context-aware view may be carried out manually or semi-automatically by composing partial views corresponding to individual elements in that context [10]. In this paper, a context is specified in a similar manner as in [8], but with a different purpose. The main purpose in [8] is size reduction (i.e., separating useful data from noise in a given context); in our work however, the main purpose is quality-based selection (i.e., selecting a subset of data that best meets certain quality requirements).

On the use of quality assessments for query answering, one of the most relevant works is [24]. Naumann's proposal is based on a universal relation [23] constructed from the global relational schema for integrating autonomous data sources. Queries are a set of attributes from the universal relation with possible value conditions over the attributes. To map a query to source views, user queries are translated to queries against the global relational schema. Naumann defines several quality criteria to qualify the sources, such as believability, objectivity, reputation and verifiability, among others. These criteria are then used to define a quality model for query plans.

According to [24], the quality of a query plan is determined as follows. Each source receives information quality (IQ) scores for each criterion considered relevant, which are then combined in an IQ-vector where each component corresponds to a different criterion. Users can specify their preferences of the selected criteria by assigning weights to the components of the IQ-vector, hence obtaining a weighting vector. This weighting vector is used in turn by multi-attribute decision-making (MADM) methods for ranking the data sources participating in the universal relation. These methods range from the simple scaling and summing of the scores (SAW) to complex formulas based on concordance and discordance matrices. The quality model is independent of the MADM method chosen, as long as it supports user weighting and IQ-scores. Given IQ-vectors of sources, the goal is to obtain the IQ-vector of a plan containing the sources. Plans are described as trees of joins between the sources: leaves are sources whereas inner nodes are joins. IQ-scores are computed for each inner node bottom-up and the overall quality of the plan is given by the IQ-score of the root of the tree.

There has been some work on the formalization and use of contexts done by the knowledge representation community. There are general, high level ideas in that line of research that are are shared with our work, namely, the idea of integration and interoperability of models and theories. In [15], the emphasis is placed on the proper interaction of different logical environments. More recently, the notion of context, or better, multi-contexts, has been formalized through the use of *bridge rules* between denoted contexts, each of which can have a knowledge base or ontology on its own [12]. The bridge rules are expressed as propositional logic programming rules. It is not clear that they can express the rich mappings found in data management applications. Not necessarily explicitly referring to contexts, there is also recent work on the integration of ontologies and distributed description logics (cf. [18] and references therein) that shows ideas similar to those found in the literature on contexts in knowledge representation.

6 Discussion and Conclusions

We have proposed a general framework that allows for the assessment of a database instance in terms of quality properties. These properties are captured in terms of alternative instances that are obtained by interaction with additional contextual data or metadata. Our framework involves mappings between database schemas like those found in data exchange, virtual data integration and peer data management systems (PDMSs). Quality answers to a query also become relative to the alternative instances that emerge from the interaction between the instance under assessment and the contextual data or metadata.

These are first steps in the direction of capturing data quality and quality query answering as context dependent activities. We examined a few natural cases of the general framework. We also made some assumptions about the mappings, views and queries involved. The general framework and also cases of more intricate mappings (cf. [5, 6] for more complex mappings in PDMs) remain to be investigated. More algorithms have to be proposed and investigated, both for quality assessment and for quality query answering.

Among the most prominent objects of ongoing research, we can mention: (a) The use of external quality predicates and data in the assessment of a given database instance. (b) The use and integration in our framework of more "intrinsic" and "absolute" quality predicates of the kind introduced in [19]. They can capture aspects as deep as data value syntax, correctness, sense, meaning, timeliness, etc. (c) A detailed and comparative analysis of the quality measures mentioned in this paper and others.

Contexts have appeared in the data management literature, mostly in relation with obvious contextual aspects of data, like geographic and temporal dimensions. However, in our view, a general notion of context, its formalization, and use in data management have been missing so far. This is a most important problem that still has to be fully investigated. We have proposed some first ideas in this direction.

References

1. Abiteboul, S., Hull, R. and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.
2. Ballou, D., Wang, R., Pazer, H. and Tayi, G. Modeling Information Manufacturing Systems to Determine Information Product Quality. *Management Science*, 44(4):462-484, 1998.
3. Batini, C. and Scannapieco, M. *Data Quality: Concepts, Methodologies and Techniques.* Springer, 2006.
4. Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In *Inconsistency Tolerance*, Springer LNCS 3300, 2004, pp. 42-83.
5. Bertossi, L. and Bravo, L. Query Answering in Peer-to-Peer Data Exchange Systems. In 'Current Trends in Database Technology', Springer LNCS 3268, 2004, pp. 478-485.
6. Bertossi, L. and Bravo, L. The Semantics of Consistency and Trust in Peer Data Exchange Systems. Proc. International Conference on Logic for Programming, Ar-

tificial Intelligence, and Reasoning (LPAR 07), 2007, Springer LNCS 4790, pp. 107-122.

7. Bleiholder, J. and Naumann, F. Data Fusion. *ACM Computing Surveys*, 41(1):1-41, 2008.
8. Bolchini, C., Curino, C., Orsi, G., Quintarelli, E., Rossato, R., Schreiber, F. and Tanca, L. And What Can Context Do for Data? *Communications of the ACM*, 52(11):136-140, 2009.
9. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F. and Tanca, L. A Data-Oriented Survey of Context Models. *SIGMOD Record*, 36(4):19-26, 2007.
10. Bolchini, C., Quintarelli, E. and Rossato, R. Relational Data Tailoring Through View Composition. Proc. ER'07, Springer LNCS 4801, 2007, pp. 149-164.
11. Bravo, L. and Bertossi, L. Logic Programs for Consistently Querying Data Integration Systems. *Proc. International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003, Morgan Kaufmann, pp. 10-15.
12. Brewka, G. and Eiter, Th. Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. Proc. AAAI 2007, pp. 385-390.
13. De Giacomo, G., Lembo, D., Lenzerini, M. and Rosati, R. On Reconciling Data Exchange, Data Integration, and Peer Data Management. Proc. PODS 2007. pp. 133-142.
14. Duschka, O., Genesereth, M. and Levy, A. Recursive Query Plans for Data Integration. *Journal of Logic Programming*, 2000, 43(1):49-73.
15. Giunchiglia, F. and Serafini, L. Multilanguage Hierarchical Logics. *Artificial Intelligence*, 1994, 65:29-70.
16. Grahne, G. and Mendelzon, A. O. Tableau Techniques for Querying Information Sources through Global Schemas. Proc. ICDT 1999. pp. 332-347.
17. Halevy, A. Answering Queries Using Views: A Survey. *VLDB Journal*, 2001, 10(4):270-294.
18. Homola, M. and Serafini, L. Towards Formal Comparison of Ontology Linking, Mapping and Importing. In Proc. DL'10, CEUR-WS 573, 2010, pp. 291-302.
19. Jiang, L., Borgida, A. and Mylopoulos, J. Towards a Compositional Semantic Account of Data Quality Attributes. Proc. ER'08, Springer LNCS 5231, 2008, pp. 55-68.
20. Kivinen, J. and Mannila, H. Approximate Inference of Functional Dependencies from Relations. *Theoretical Computer Science*, 1995, 149:129-149.
21. Kolaitis, Ph. Schema Mappings, Data Exchange, and Metadata Management. Proc. PODS'05, 2005, pp. 61-75.
22. Lenzerini, M. Data Integration: A Theoretical Perspective. Proc. PODS'02, 2002, pp. 233-246.
23. Maier, D., Ullman, J. and Vardi, M. On the Foundations of the Universal Relation Model. *ACM Transactions on Database Systems*, 1984, 9(2):283-308.
24. Naumann, F. *Quality-Driven Query Answering for Integrated Information Systems.* Springer, 2002.
25. Wang, R. and Strong, D. Beyond Accuracy: What Data Quality Means to Data Consumers. *J. Management and Information Systems*, 1996, 12(4):5-33.