Ontology-Based Multidimensional Contexts with Applications to Quality Data Specification and Extraction

Mostafa Milani and **Leopoldo Bertossi**
Carleton University, School of Computer Science
Ottawa, Canada

Abstract. Data quality assessment and data cleaning are context dependent activities. Starting from this observation, in previous work a context model for the assessment of the quality of a database was proposed. A context takes the form of a possibly virtual database or a data integration system into which the database under assessment is mapped, for additional analysis, processing, and quality data extraction. In this work, we extend contexts with dimensions, and by doing so, multidimensional data quality assessment becomes possible. At the core of multidimensional contexts we find ontologies written as Datalog$^{\pm}$ programs with provably good properties in terms of query answering. We use this language to represent dimension hierarchies, dimensional constraints, dimensional rules, and specifying quality data. Query answering relies on and triggers dimensional navigation, and becomes an important tool for the extraction of quality data.

1 Introduction

Data quality assessment and data cleaning are context-dependent activities. More precisely, the quality of data has to be assessed with some form of contextual knowledge, in particular, about the *production and the use* of data, among other possible dimensions of data quality. Data quality refers to the degree to which data fits or fulfills a form of usage [3, 23]. As expected, context-based data quality assessment requires a formal model of context. Accordingly, we propose a model of context that addresses quality concerns that are related to the production and use of data.

Here we follow and extend the approach in [4] that provides a model of context for data quality assessment. In that work, the assessment of a database D is performed by *putting D in context*, more precisely, by mapping it into a context $\mathcal{C}$ (Fig. 1, left), which is represented as another database, or as a database schema with partial information, or, more generally, as a virtual data integration system [25]. The latter may have some materialized data and access to external data sources.

The quality of data in D is determined through additional processing, material or virtual, of the data within the context. These contextual data may be imported from D or may be already available at the context. The context may also contain application-dependent knowledge associated to data quality, in the form of rules or semantic constraints. Data processing in the context leads to possibly several quality versions of D, forming a class $\mathcal{D}^q$ of intended, clean versions of D (Fig. 1, right). The quality of D is measured in terms of how much D departs from (its quality versions in) $\mathcal{D}^q$: $dist(D, \mathcal{D}^q)$. Of course, different distance measures may be used for this purpose [4].

In some cases, we may want to assess the quality of answers to a query $\mathcal{Q}$ posed to instance D or to obtain "quality answers" from D. This can be done appealing to the class $\mathcal{D}^q$ of intended clean versions of D. For assessment, the set of query answers to

$\mathcal{Q}$ from D can be compared with the *certain answers* for $\mathcal{Q}$, i.e. the intersection of the sets of answers to $\mathcal{Q}$ from each of the instances in $\mathcal{D}^q$ [22]. The certain answers become what we could call the *clean answers* to $\mathcal{Q}$ from D [4]. So, if we want the clean answers to $\mathcal{Q}$ from D, instead of computing the answers from D as usual, we compute the clean answers (cf. right-hand side of Fig. 1).

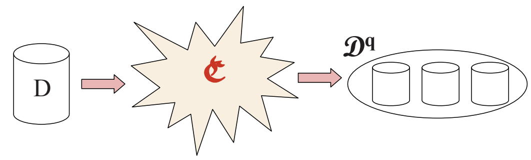

Fig. 1. Clean instances and query answers

When computing clean query answers, instead of computing, materializing and querying all the instances in class $\mathcal{D}^q$, a form of *query rewriting* can be attempted: a new query $\mathcal{Q}^q$ is posed to D to obtain the clean answers for $\mathcal{Q}$. Some cases of rewriting were investigated in [4]. In this work we continue adopting this approach to data quality assessment and clean query answering. However, as we will see, the contexts we consider in this work are more complex than those considered in [4], and for good reasons.

An important contextual element was not considered in [4]: *dimensions*. They were *not* considered as contextual elements for data quality analysis, but in practice, dimensions are naturally associated to contexts. Here, in order to capture general dimensional aspects of data for inclusion in contexts, we take advantage of and start from the Hurtado-Mendelzon (HM) multidimensional data model [21], whose inception was mainly motivated by data warehouses (DWH) and OLAP applications.

We extend the HM model by adding *categorical relations* associated to categories, at different levels of the dimension hierarchies, possibly to more than one dimension (think of generalized fact tables as found in data warehouses). It also includes *dimensional constraints* and *dimensional rules*, which could be treated both as *dimensional integrity constraints* on categorical relations that involve values from dimension categories. However, dimensional constraints are intended to be used as *denial constraints* that forbid certain combinations of values, whereas the dimensional rules are intended to be used for data completion, to generate data through their enforcement via *dimensional navigation*.

In this work we propose an ontological representation in Datalog$^\pm$ [8, 9] of the extended HM model, and also mechanisms for data quality assessment based on query answering from the ontology via dimensional navigation. As already suggested, the idea is that a query to the ontology triggers dimensional navigation and the creation of missing data, in possible upward and downward directions, and on multiple dimensions. Datalog$^\pm$ supports data generation through the ontological rules. This is particularly useful, and also much in line with the way we understand and use contexts in everyday life: *Context allows us to extend or expand information that, otherwise, without this extension, would be impossible or difficult to understand or make sense of*. Furthermore, this ontological approach captures well our general philosophy according to which, *contexts should be represented as formal theories into which other objects, like database instances, are mapped*, for contextual analysis, assessment, interpretation, and additional processing [4].

Datalog$\pm$ is an extension of classical Datalog, mainly through the use of existentially quantified variables (a.k.a. value invention) in rule heads. It has been successfully applied to the logical representations of data models and ontologies [11, 13]. Actually, a *multidimensional (MD) context* corresponding to the formalization of the extension of HM becomes a Datalog$\pm$ ontology, $\mathcal{M}$, that belongs to an interesting syntactic class of programs, for which some results are known. This allows us to give a semantics to our ontologies, and apply some established and new algorithms for query answering.

More precisely, the core MD ontology $\mathcal{M}$ is a weakly-sticky Datalog$^\pm$ program [12], for which (conjunctive) query answering has polynomial-time data complexity. In our case, weak-stickiness is due to the as we argue, natural assumptions that: (a) dimension navigation (as captured by data generation) happens through rules with body joins on *categorical attributes* (i.e. in categorical relations), whose values come from dimension categories; and (b) there is no value invention for categorical attributes. (We also discuss cases where these assumptions do not hold.)

MD ontologies are used to support quality data specification and extraction. [1] More precisely, and continuing with the above idea on this use of contexts, it amounts to: (a) defining application-dependent *quality predicates* (they can be seen as views capturing data quality concerns), (b) using them to define the *quality versions* of the original predicates (relations) in the database D under quality assessment, and (c) retrieving quality data by querying the (possibly virtual extensions of the) latter predicates [4]. These predicate definitions may be based on *data quality guidelines* that are captured as rules or semantic constraints, both of which may refer to categorical attributes of predicates in $\mathcal{M}$, without being part of $\mathcal{M}$. Rather, this "quality part" of the context comes on top of $\mathcal{M}$. We establish that under reasonable conditions on these extra definitions, the resulting extension of $\mathcal{M}$ still retains the tractability of query answering (even when weak-stickiness may be compromised).

About related work, in [6] dimensions become the basis for *building* contexts, or more precisely database instances that are tailored according to certain dimensional elements. This is done through a process of selection of relevant dimensional elements: the dimension leaves a footprint on the data. As a result, the constructed database is implicitly dimensional, and the dimensions as such may be lost as first-class objects in the generated context.

In [27, 28] the authors consider the generation of data at different levels of a category hierarchy, and at query answering time. This involves hierarchy navigation and an extension of relational algebra that computes data by appealing to data at other levels of the hierarchy. Actually, in our work we show how this process can be captured via our Datalog$^\pm$ MD ontologies.

DWHs have been represented in expressive description logics (DL) [16, 17]. Preliminary research on extensions in DL of the HM model, also for data quality purposes, can be found in [24].

Summarizing, in this work we make the following contributions: [2]

1. We extend the HM data model and represent the extension as a Datalog$^\pm$ ontology that contains: (a) categorical relations, (b) tuple-generating-dependencies, *tgds* (a

[1] In this work we do not explicitly address the problem of assessing the quality of the original data through a numerical comparison with the quality data [4].

[2] This work considerably extends [29], which contains basically the material of Section 2 here.

rule incarnation of referential constraints), to connect the original data to categorical relations, and the latter to dimensions; and (c) dimensional constraints.

2. We establish that the MD ontology is a *weakly-sticky* Datalog$^\pm$ program [12]. As a consequence, query answering can be done in polynomial time.
3. We analyze the effect of dimensional constraints on query answering, specifically the *separability condition* [12] between *tgds* and constraints that are equality-generating-dependencies, *egds*. We show that by restricting variables in equalities to appear categorical attributes, separability holds.
4. We propose a general approach for contextual data quality specification and extraction that is based on MD ontologies, emphasizing the dimensional navigation process that is triggered by queries about quality data. We illustrate the application of this approach by means of an extended example.

2 An Extended, Motivating Example

This section illustrates the intuition behind categorical relations, dimensional rules and constraints, and how they are used for data quality purposes. We assume, according to the HM model (cf. Section 3), that a dimension consists of a finite set of categories related to each other by a partial order.

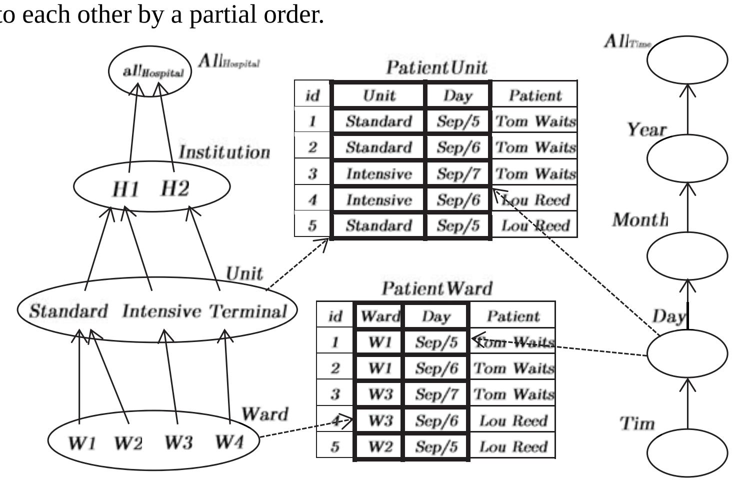

Fig. 2. An extended multidimensional model

Example 1. The relational table *Measurements* (Table 1) shows body temperatures of patients in an institution. A doctor wants to know "*The body temperatures of Tom Waits for September 5 taken around noon with a thermometer of brand B1*" (as he expected). Possibly a nurse, unaware of this requirement, used a thermometer of brand *B2*, storing the data in *Measurements*. In this case, not all the measurements in the table are up to the expected quality. However, table *Measurements* alone does not discriminate between intended values (those taken with brand *B1*) and the others.

For assessing the quality of the data in *Measurements* according to the doctor's quality requirement, extra contextual information about the thermometers in use may help. In this case, the contextual information is in table *PatientWard*, linked to the *Ward* category (Fig. 2, middle, bottom). This *categorical relation* stores patient names for each ward of the institution.

Furthermore, the institution has a *guideline* prescribing that: *"Temperature measurement for patients in a standard care unit have to be taken with thermometers of Brand B1"*. It can be used for data quality assessment when combined with categorical table *PatientUnit* (Fig. 2, middle, top), which is linked to the *Unit* category, and whose data are (at least partially) generated from *PatientWard* by upward-navigation through dimension Hospital (Fig. 2, left), from category *Ward* to category *Unit*.

Table 1. *Measurements*

	Time	Patient	Value
1	Sep/5-12:10	Tom Waits	38.2
2	Sep/6-11:50	Tom Waits	37.1
3	Sep/7-12:15	Tom Waits	37.7
4	Sep/9-12:00	Tom Waits	37.0
5	Sep/6-11:05	Lou Reed	37.5
6	Sep/5-12:05	Lou Reed	38.0

Table 2. $Measurements^q$

	Time	Patient	Value
1	Sep/5-12:10	Tom Waits	38.2
2	Sep/6-11:50	Tom Waits	37.1

According to the guideline, it is now possible to conclude that, on days when Tom Waits was in the standard care unit, his temperature values were taken with the expected thermometer: for patients in wards *W1* or *W2* a thermometer of brand *B1* was used. These "clean data" in relation to the doctor's expectations appear in relation $Measurements^q$ (Table 2).

Elaborating on this example, there could be a *dimensional constraint*: *"No patient in intensive care unit at any time during August 2005"*. As stated, this constraint could be represented as a "static" constraint on the categorical relation *PatientUnit*. However, it could also be represented as one on the data generation process via upward-navigation from *PatientWard* to *PatientUnit*, preventing the use of the third tuple in table *PatientWard*. As such, this becomes a *navigational constraint* that also involves dimensions Hospital and Time (Fig. 2, right). A third alternative is handling the constraint as a "static" constraint on the join of *PatientWard* and *PatientUnit* via the patient name (*Tom Waits* could not be both in ward *W3* and intensive care on some dates). Our approach will allow to handle the constraint in any of these three forms. ■

Categorical relations may be incomplete, and new data can be generated for them, which will be enabled through rules (*tgds*) of a Datalog± dimensional ontology. The previous example shows data generation via upward navigation. Our next example shows that *downward navigation* may also be useful. Our approach to MD contexts will support both.

Example 2. (ex. 1 cont.) Consider two additional categorical relations, *WorkingSchedules* (Table 3) and *Shifts* (Table 4), linked to categories *Unit* and *Ward*, resp. They store schedules of nurses in units and shifts of nurses in wards, resp. A query to *Shifts* asks for dates when *Mark* was working in ward *W2*, which has no answer with the data in Table 4. A new guideline states: *"If a nurse works in a unit on a specific day, he/she has shifts in every ward of that unit on the same day"*. It can be captured as a dimensional rule connecting *WorkingSchedules* to *Shifts* via the dimension hierarchy. Downward data generation using this rule, tuple 5 in Table 3, and the dimensional connection of *Standard* to *W1*, *W2*, makes *Mark* have shifts in both *W1* and *W2* on *Sep/9*. ■

Table 3. *WorkingSchedules*

	Unit	Day	Nurse	Type
1	Intensive	Sep/5	Cathy	cert.
2	Standard	Sep/5	Helen	cert.
3	Standard	Sep/6	Helen	cert.
4	Terminal	Sep/5	Susan	non-c.
5	Standard	Sep/9	Mark	non-c.

Table 4. *Shifts*

	Ward	Day	Nurse	Shift
1	W4	Sep/5	Cathy	night
2	W1	Sep/6	Helen	morning
3	W4	Sep/5	Susan	evening

3 Preliminaries

Contextual Data Quality: We first briefly review previous work in [4] on context-based data quality assessment. The starting point is that *data quality is context dependent*. A context provides *knowledge about the way data are interrelated, produced and used*, which allows us to make sense of the data. In our view, both the database under quality assessment and the context can be formalized as logical theories. The former is then *put in context* by mapping it into the latter, through logical mappings and possibly shared predicates.

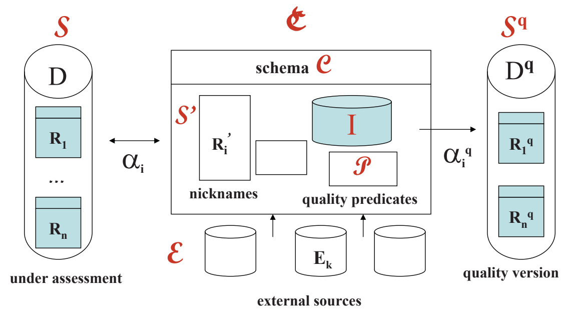

Fig. 3. A context for data quality assessment

In Fig. 3, D is a relational database (with schema $\mathcal{S}$) under quality assessment. It can be represented as a logical theory [32]. The context, $\mathfrak{C}$ in the middle, resembles a virtual data integration system, which can also be represented as a logical theory [25]. The context has a relational schema (or signature), $\mathcal{C}$, in particular predicates with possibly partial extensions (incomplete relations). The mappings between $\mathcal{C}$ and D are of the kind used in data integration or data exchange [19], that can be expressed as logical formulas. In this paper, we are not concerned with how such a context is created [4].

A subschema of $\mathcal{C}$ may have an instance I, but $\mathcal{C}$ has nicknames (copies) R' for predicates R in $\mathcal{S}$. Nicknames are used to map (via α_i) the data in D into $\mathfrak{C}$, for further logical processing. So, schema $\mathcal{C}$ can be seen as an expansion of $\mathcal{S}$ through a subschema $\mathcal{S}'$. Some predicates in $\mathcal{C}$ are meant to be *quality predicates* (in $\mathcal{P}$), which are used to specify single quality requirements. There may be semantic constraints on schema $\mathcal{C}$, and also access (mappings) to external data sources, in $\mathcal{E}$, that could be used for data assessment or cleaning.

A clean version of D, obtained through the mapping into and processing within context $\mathfrak{C}$, is a possibly virtual instance D^q (or a collection thereof, as suggested in Fig. 1), for schema $\mathcal{S}^q$ (a "quality" copy of schema $\mathcal{S}$). The extension of every predicate in it, say R^q, is the "quality version" of relation R in D, and is defined as a view (via the α_i^q) in terms of the nickname predicates in $\mathcal{S}'$, those in $\mathcal{P}$, and other contextual predicates. The quality of (the data in) instance D can be measured by comparing D with the instance D^q or the set, $\mathcal{D}^q$, of them. This latter set can also be used to define and possibly compute the *quality answers* to queries originally posed to D, as the *certain answers* w.r.t. $\mathcal{D}^q$. See [4] for more details, and different cases that may occur. In any case, the main idea is that quality data can be extracted from D by querying the possibly virtual class $\mathcal{D}^q$.

In this paper, we extend the approach to data quality specification and extraction we just described, by adding dimensions to contexts, for multidimensional data quality specification and extraction. In this case, the context contains a generic MD ontology, the shaded $\mathcal{M}$ in Fig. 4, a.k.a. "core ontology" (and described in Section 4). This ontology can be extended, within the context, with additional rules and constraints that depend on specific data quality concerns (cf. Section 6).

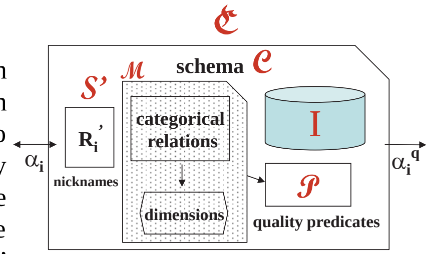

Fig. 4. A multidimensional context

The Hurtado-Mendelzon Data Model: According to the Hurtado-Mendelzon (HM) multidimensional data model [21], a *dimension schema*, $\mathcal{S} = \langle \mathcal{K}, \nearrow \rangle$, is a directed acyclic graph and lattice, with $\mathcal{K}$ a set of categories (represented as unary predicates), and $\nearrow$ the *parent-child relation* between categories. $\nearrow^*$ denotes the transitive and reflexive closure of $\nearrow$, and is a partial order with a *top category*, *All*, which is reachable from every other category. There is a unique *base category*, which does not have children. A *dimension instance* for schema $\mathcal{S}$ is a tuple $\mathcal{D} = \langle \mathcal{N}, <, \sigma \rangle$, with $\mathcal{N}$ a set of *elements*, $<$ is a *parent-child relation* between elements, and $\sigma : \mathcal{N} \rightarrow \mathcal{K}$, the *membership function*, is total and injective. A dimension instance is shown in Fig. 2, left. The partial order $<$ parallels (is consistent with) $\nearrow$: $a < b$ implies $\sigma(a) \nearrow \sigma(b)$. $\sigma(e) = k$ is also denoted as $e \in k$ or $k(e)$ (holds). $<^*$ is the transitive and reflexive closure of $<$, and is used to define the *roll-up* relations for any pair of categories k and k': $L_k^{k'}(\mathcal{D}) = \{(e, e') \mid e \in k,\ e' \in k' \text{ and } e <^* e'\}$.

Datalog$^\pm$: Datalog$^\pm$ [8, 9] is a family of rule languages that properly extends plain Datalog with: (a) rules (*tgds*) may have existential quantifiers in the heads; (b) *equality-generating dependencies* (*egds*), i.e. rules with only equality in the head; and (c) *negative constraints* (NCs), that are rules with $\bot$, a false propositional atom, in the heads, indicating that the rule body cannot be true.

Example 3. This Datalog$^\pm$ program shows a *tgd*, an *egd*, and an *NC*, in this order: $\exists x Assist(d, x) \leftarrow Doctor(d)$; $x = x' \leftarrow Assist(d, x), Assist(d, x')$; $\bot \leftarrow Specialist(d, x, n), Nurse(d, n)$. ■

Datalog$^\pm$ has been used to represent ontological knowledge and conceptual data models [11, 13]; and for *ontology-based data access* [15, 18]. The underlying extensional, relational database (the facts) $\mathcal{I}$ for a program may be incomplete, and the *chase* is the standard procedure for completing the database, through the enforcement of the program rules. When a *tgd* is applied, new atoms are created, possibly including fresh nulls (for the existential variables), and the whole run of the chase may be non-terminating, leading to an infinite complete database. The enforcement of an *egd* equates nulls with nulls or nulls with constants or fails. For a set Σ of *tgds* and *egds*, $chase(\mathcal{I}, \Sigma)$ denotes the possibly infinite instance resulting from the non-failing chase of Σ on $\mathcal{I}$.

Even with an infinite $chase(\mathcal{I}, \Sigma)$ it is possible that *conjunctive query answering* (QA) is decidable (or computable). The $^-$ in Datalog$^\pm$ stands for syntactic restrictions on the interaction of *tgds* in Σ that ensure decidability of QA, and, in some cases, also tractability (in data). Datalog$^\pm$ is a family of languages with different degrees of expressivity and computational properties. Some of them are: *linear*, *guarded*, *weakly-guarded*, *sticky*, and *weakly-sticky* Datalog$^\pm$ [8, 9, 10, 11, 12]. In this work (cf. [31, appendix A]), we are particularly interested in *weakly-sticky* (WS) Datalog$^\pm$ [12], which extends *sticky* Datalog$^\pm$ [10].

4 Extending the HM Model with Datalog$^\pm$

We extend the HM model introducing *categorical relations*, each of them having a relational schema with a name, and attributes, some of which are *categorical* and the other, *non-categorical*. The former take values that are members of a dimension category. The latter take values from an arbitrary domain. Categorical relations have to be logically connected to dimensions. For this we use a Datalog± ontology $\mathcal{M}$, which has a relational schema $\mathcal{S}_\mathcal{M}$, an instance $\mathcal{D}_\mathcal{M}$, and a set $\Sigma_\mathcal{M}$ of dimensional rules, and a set $\kappa_\mathcal{M}$ of constraints. Here, $\mathcal{S}_\mathcal{M} = \mathcal{K} \cup \mathcal{O} \cup \mathcal{R}$, with $\mathcal{K}$ a set of unary *category predicates*, $\mathcal{O}$ a set of *parent-child predicates*, capturing $<$-relationships for pairs of adjacent categories, and $\mathcal{R}$ a set of *categorical predicates*, say $R(C_1, \ldots; N_1, \ldots)$, where, to highlight, categorical and non-categorical attributes (C_is vs. N_js) are separated by ";".

Example 4. Categorical relation $PatientWard(Ward{,}Day;Patient)$ in Fig. 2 has categorical attributes *Ward* and *Day*, connected to the Hospital and Time dimensions, resp. *Patient* is non-categorical. $Ward(\cdot)$, $Unit(\cdot) \in \mathcal{K}$; $\mathcal{O}$ contains, e.g. a binary predicate connecting *Ward* to *Unit*; and $\mathcal{R}$ contains, e.g. *PatientWard*. ■

The (extensional) data, $\mathcal{D}_\mathcal{M}$, associated to the ontology $\mathcal{M}$'s schema are the complete extensions for categories in $\mathcal{K}$ and predicates in $\mathcal{O}$ that come from the dimension instances. The categorical relations (with predicates in $\mathcal{R}$) may contain partial data, i.e. they may be incomplete. They can belong to instance I in Fig. 4. Dimensional rules in $\Sigma_\mathcal{M}$ are those in (c) below; and constraints in $\kappa_\mathcal{M}$, those in (a) and (b).

(a) *Referential constraints* between categorical attributes and categories as negative constraint:[3] ($R \in \mathcal{R}$, $K \in \mathcal{K}$; $\bar{e}, \bar{a}$ are categorical, non-categorical, resp.; $e \in \bar{e}$)

$$\bot \;\leftarrow\; R(\bar{e};\bar{a}), \neg K(e). \tag{1}$$

Notice that K, to which negation is applied, is a closed, extensional predicate.

(b) Additional *dimensional constraints*, as *egds* or *NCs*: ($R_i \in \mathcal{R}$, $D_j \in \mathcal{O}$, and x, x' stand both for either categorical or non-categorical attributes in the body of (2))

$$x = x' \;\leftarrow\; R_1(\bar{e}_1;\bar{a}_1), ..., R_n(\bar{e}_n;\bar{a}_n), D_1(e_1, e_1'), ..., D_m(e_m, e_m'). \tag{2}$$

$$\bot \;\leftarrow\; R_1(\bar{e}_1;\bar{a}_1), ..., R_n(\bar{e}_n;\bar{a}_n), D_1(e_1, e_1'), ..., D_m(e_m, e_m'). \tag{3}$$

(c) *Dimensional rules* as Datalog$^\pm$ *tgds*:

$$\exists \bar{a}_z\, R_k(\bar{e}_k;\bar{a}_k) \leftarrow R_1(\bar{e}_1;\bar{a}_1), ..., R_n(\bar{e}_n;\bar{a}_n), D_1(e_1, e_1'), ..., D_m(e_m, e_m'). \tag{4}$$

Here, $\bar{a}_z \subseteq \bar{a}_k$, $\bar{e}_k \subseteq \bar{e}_1 \cup ... \cup \bar{e}_n \cup \{e_1, ..., e_m, e_1', ..., e_m'\}$, $\bar{a}_k \smallsetminus \bar{a}_z \subseteq \bar{a}_1 \cup ... \cup \bar{a}_n$; and repeated variables in bodies are only in positions of categorical attributes

[3] An alternative and more problematic approach, may use *tgds* between categorical attributes and categories, making it possible to generate elements in categories or categorical attributes.

(in the categorical relations $R_i(\bar{e}_i;\bar{a}_i)$), and attributes in parent-child predicates $D_j(e_j, e'_j)$[4]. Value invention is only on non-categorical attributes (we will consider relaxing this later on).

Some of the lists in the bodies of (2)-(4) may be empty, i.e. $n = 0$ or $m = 0$. This allows us to represent, in addition to properly "navigational" constraints, also classical constraints on categorical relations, e.g. keys or FDs.

Example 5. (ex. 1 and 4 cont.) In relation *PatientUnit*, the categorical attribute *Unit* takes values from the *Unit* category. We use a constraint of the form (1), namely: $\bot \leftarrow PatientUnit(u, d; p), \neg Unit(u)$. The constraint *"No patient in intensive care unit during August 2005"* becomes a dimensional (navigational) constraint of the form (3):

$$\bot \leftarrow [PatientWard(w, d; p),\ UnitWard(\texttt{Intensive}, w),\ MonthDay(\texttt{August2005}, d)]. \tag{5}$$

Alternatively, we could apply a constraint directly on $PatientUnit$, without explicit navigation in the Hospital dimension, but we still need to navigate in the Time dimension: $\bot \leftarrow PatientUnit(\texttt{Intensive}, d; p), MonthDay(\texttt{August2005}, d)$.

An *egd* of the form (2) says that *"All thermometers in a unit are of the same type"*:

$$t = t' \leftarrow Therm(w, t; n),\ Therm(w', t'; n'), UnitWard(u, w), UnitWard(u, w') \tag{6}$$

with $Therm(\,Ward, Thertype; Nurse)$ a categorical relation, and *Ward*, *Thertype* categorical attributes (the latter for an Instrument dimension). This *egd* illustrates the flexibility of our approach. Even without having a categorical relation at the *Unit*, we could still impose a condition at that level.[5]

The following *tgds* generate data from *PatientWard* to *PatientUnit*, and from *WorkingSchedules* to *Shifts*, resp. They are of the form (4).

$$PatientUnit(u, d; p) \ \leftarrow\ PatientWard(w, d; p),\ UnitWard(u, w). \tag{7}$$

$$\exists z\ Shifts(w, d; n, z) \ \leftarrow\ WorkingSchedules(u, d; n, t),\ UnitWard(u, w). \tag{8}$$

The existential variable in (8) makes up for the missing, non-categorical attribute in the "parent" relation *WorkingSchedules*. This is not needed in (7). ■

Remark 1. A general *tgd* of the form (4) enables *upward-* or *downward-navigation*, depending on the body joins. The direction is determined by the dimension levels of categorical attributes in the joins. For simplicity, assume that there is a single $D_j \in \mathcal{O}$ in the body (as in (7) and (8)). If the join is between $R_i(\bar{e}_i;\bar{a}_i)$ and $D_j(e_j, e'_j)$ then: (a) (one-step) upward navigation is enabled, from e'_j to e_j, when $e'_j \in \bar{e}_i$ (i.e. e'_j appears in $R_i(\bar{e}_i;\bar{a}_i)$) and $e_j \in \bar{e}_k$, i.e in the head), (b) (one-step) downward navigation is enabled, from e_j to e'_j, when e_j occurs in R_i and e'_j occurs in R_k. Several occurrences of parent-child predicates in a body capture multi-step navigation. ■

[4] This is a natural restriction since dimension navigation is captured by the joins only between variables of these attributes

[5] If we have that relation, as in Example 1, then (6) could be replaced by a "static", non-navigational FD. This issue is further discussed in [31, appendix B].

Example 6. (ex. 5 cont.) Rule (8) captures downward-navigation; and this is a general behavior with *tgds* of the form (4). That is, when drilling-down via (8), from a tuple, say $WorkingSchedules(u, d; n, t)$ via the category member u (for Unit), *for each* child w of u in the *Ward* category, a tuple for *Shifts* is generated, as specified in the body of (8). For example, chasing (8) with the last tuple in Table 3, generates the new tuple $\langle \texttt{W1}, \texttt{Sep/9}, \texttt{Mark}, \perp \rangle$ in Table 4, with a fresh null for the shift (similarly for *W2*). This allows us to answer the query about the dates *Mark* works in *W1*: $\mathcal{Q}'(d)\!:\ \exists s Shifts(\texttt{W1}, d, \texttt{Mark}, s)$. We obtain *Sep/9*.

Instead, the join between *PatientWard* and *UnitWard* in (7) enables upward-dimension navigation; and generates only one tuple for *PatientUnit* from each tuple in *PatientWard*, because each *Ward* member has only one *Unit* parent. ■

5 Properties of MD Datalog$^\pm$ Ontologies

Here, we first establish the membership of our MD ontologies, $\mathcal{M}$ (cf. Section 4) of a class of the Datalog± family. Membership is determined by the set $\Sigma_\mathcal{M}$ of its *tgds*. Next, we analyze the role of the constraints in $\kappa_\mathcal{M}$, in particular, of the set $\epsilon_\mathcal{M}$ of *egds*.

Proposition 1. MD ontologies are weakly-sticky Datalog± programs. ■

The proof (as other proofs) and a review of *weakly-sticky* Datalog± [12] can be found in the extended version [31, appendix A.]. A consequence of this result is that conjunctive query answering (QA) from $\Sigma_\mathcal{M}$ is in polynomial-time in data complexity [12]. The complexity stays the same if we add negative constraints, *NCs*, of the forms (1) and (3), because they can be checked through the conjunctive queries in their bodies [12]. However, combining the *egds* in $\epsilon_\mathcal{M}$ with $\Sigma_\mathcal{M}$ could change things, and, in principle, even lead to undecidability of QA [7].

Example 7. Consider $\mathcal{I} = \{Surgery(\texttt{W1}, \texttt{John})\}$ and a weakly-sticky set Σ_T of *tgds*: $\sigma_1 : \exists z\ Surgeon(w, z) \leftarrow Surgery(w, p)$; $\sigma_2 : \exists y\ Assist(w, y) \leftarrow Surgery(w, p)$; $\sigma_3 : \exists z\ Surgery(z, x) \leftarrow Assist(w, x), Surgeon(w', x)$. Here, $chase(\mathcal{I}, \Sigma_T) = \{Surgery(\texttt{W1}, \texttt{John}), Assist(\texttt{W1}, \perp_1), Surgeon(\texttt{W1}, \perp_2)\}$.

Now, if we add the *egd* ε: $y = z \leftarrow Assist(w, z), Surgeon(w, y)$, the chase is infinite: $chase(\mathcal{I}, \Sigma_T \cup \{\varepsilon\}) = \{Surgery(\texttt{W1}, \texttt{John}), Assist(\texttt{W1}, \perp_1), Surgeon(\texttt{W1}, \perp_1), Surgery(\perp_2, \perp_1), Assist(\perp_2, \perp_3), Surgeon(\perp_2, \perp_3), Surgery(\perp_4, \perp_3), \ldots\}$.

These non-failing chases give different answers to the Boolean conjunctive query (BCQ) $\mathcal{Q}\!: \exists w x w'(Assist(w; x) \wedge Surgeon(w'; x))$: $chase(\mathcal{I}, \Sigma_T \cup \{\varepsilon\}) \models \mathcal{Q}$, but $chase(\mathcal{I}, \Sigma_T) \not\models \mathcal{Q}$. ■

This example shows a harmful interaction between the *tgds* and an *egd*. They infinitely fire each other, making infinite an initially finite chase. The interaction also has an effect on QA. A *separability condition* on the combination of *egds* and *tgds* guarantees a harmless interaction w.r.t. QA.

Definition 1. [11, 14] Let Σ be formed by a set Σ_T of *tgds* and a set Σ_E of *egds*. Σ_E and Σ_T are *separable* if, for every instance $\mathcal{I}$ for which the chase of Σ on $\mathcal{I}$ does not fail, and BCQ $\mathcal{Q}$, $chase(\mathcal{I}, \Sigma) \models \mathcal{Q}$ if and only if $chase(\mathcal{I}, \Sigma_T) \models \mathcal{Q}$. ■

Example 7 shows a case of non-separability. Separability tells us that we can safely ignore Σ_E for QA. More precisely, if separability holds and QA is decidable under the *tgds*, then it is also decidable under the combination of *tgds* and *egds* : (a) (combined) chase failure can be decided by posing conjunctive queries associated to the bodies of the *egds* [14, theo. 1]; (b) if it does not fail, QA can be done with the *tgds* alone. Even more, under separability, the complexity of QA on $\mathcal{I} \cup \Sigma$ is the same as for $\mathcal{I} \cup \Sigma_T$ [11, 13, 14].

Proposition 2. For an MD ontology $\mathcal{M}$ with a set $\Sigma_{\mathcal{M}}$ of *tgds* as in (4) and set $\epsilon_{\mathcal{M}}$ of *egds* as in (2), separability holds if, for every *egd* in $\epsilon_{\mathcal{M}}$, the variables in the equality (in the head) occur in categorical positions in the body. ■

In combination with Proposition 1, we obtain:

Corollary 1. Under the hypothesis of Proposition 2, QA from an MD ontology can be done in polynomial-time in data. ■

Under the hypothesis of Proposition 2, our MD ontologies are separable and enjoy the good properties we just mentioned. However, some good properties can still be preserved with non-separable MD ontologies. The next example motivates this result.

Example 8. (ex. 7 cont.) Let us modify our ontology. Now, $\Sigma'_T = \{\sigma_1, \sigma_2\}$, and the *egd* is still ε. Now, both chases are finite: $chase(\mathcal{I}, \Sigma'_T \cup \{\varepsilon\})$ $= \{Surgery(\mathtt{W1}; \mathtt{John}),\ Assist(\mathtt{W1}; \bot_1),\ Surgeon(\mathtt{W1}; \bot_1)\}$; and $chase(\mathcal{I}, \Sigma'_T) =$ $\{Surgery(\mathtt{W1}; \mathtt{John}),\ Assist(\mathtt{W1}; \bot_1),\ Surgeon(\mathtt{W1}; \bot_2)\}$. (As before, we use ";" to separate categorical from non-categorical attributes.) The *egd* is not separable from the *tgds*. Actually, for the same query $\mathcal{Q}$ of Example 7, and the non-failing chases, it holds: $chase(\mathcal{I}, \Sigma'_T \cup \{\varepsilon\}) \models \mathcal{Q}$, but $chase(\mathcal{I}, \Sigma'_T) \not\models \mathcal{Q}$. ■

In this example, despite the lack of separability, the application of *egds* does not trigger new *tgds* during the chase (as happens in Example 7). This is due (cf. Lemma 1 below) to the fact that $\Sigma'_T \cup \{\varepsilon\}$ respects a condition imposed on our MD ontologies: joins in *tgd* bodies only between categorical attributes. (The ontology in Example 7 had σ_3, which violates this condition.) Lemma 1 below tells us that with MD ontologies, applying *egd* chase steps does not increase the number of *tgd* chase steps. [6]

Lemma 1. For an MD ontology $\mathcal{M}$ with a set $\Sigma_{\mathcal{M}}$ of *tgds* as in (4) and a set $\epsilon_{\mathcal{M}}$ of *egds* as in (2), applying an *egd* chase step does not cause any new application of a ground *tgd*, i.e. a *tgd* body ground instantiation that did not appear without the *egds*. ■

With weakly-sticky sets of *tgds* the chase may not terminate, due to an infinite number of *tgd* chase steps. This is in particular the case for the set of *tgds* in our MD ontologies. However, QA on weakly-sticky *tgds* can be done in polynomial-time by querying an initial portion of the chase that has a polynomial *depth* [12]. By Lemma 1, if we add *egds*, QA can still be done by querying an initial portion of the chase (including *egds* now) that has the same (polynomial) *depth* as that for *tgds* alone. So, although *egds* in our MD ontologies may have an effect on QA (the two initial portions can be different), the complexity does not change w.r.t. to having only the *tgds*.

Proposition 3. For an MD ontology, QA is in polynomial-time in data complexity. ■

[6] We assume the chase, after the enforcement of a (ground) *tgd*, applies all the *egds*.

6 MD Contexts for Quality Data

We now show in general how to use a MD context, $\mathfrak{C}$, containing MD ontologies for quality data specification and extraction w.r.t. a database instance D for schema $\mathcal{S}$. We will at the same time, for illustration and fixing ideas, revisit the example in Section 2, putting it in terms of the MD context elements we presented in Section 4. Context $\mathfrak{C}$, as shown in Fig. 4, contains:

1. Nickname predicates $R' \in \mathcal{S}'$ for predicates R of original schema $\mathcal{S}$. In this case, the R' have the same extensions as in D, producing a material or virtual instance D' within $\mathfrak{C}$.

For example, $\mathit{Measurements}' \in \mathcal{S}'$ is a nickname predicate for $\mathit{Measurements} \in \mathcal{S}$, whose initial contents (in D) is under quality assessment.

2. The *core MD ontology*, $\mathcal{M}$, that includes a partial instance, $\mathcal{D}_{\mathcal{M}}$, containing dimensional, categorical data; and the Datalog$\pm$ ontology with *tgds* $\Sigma_{\mathcal{M}}$, and constraints $\kappa_{\mathcal{M}}$, among them, the *egds* $\epsilon_{\mathcal{M}}$ of Section 4. We assume that application dependent guidelines and constraints are all represented as components of $\mathcal{M}$.

In our running example, $\mathit{PatientUnit}$, $\mathit{PatientWard}$, $\mathit{WorkingSchedules}$ and $\mathit{WorkingTimes}$ are categorical relations. $\mathit{UnitWard}$, $\mathit{DayTime}$ are parent-child relations in the Hospital and Time dimensions, resp. The followings are dimensional rules (*tgds*) of $\Sigma_{\mathcal{M}}$: (with (9) a new version of (7) allowing upward-navigation in two dimensions)[7]

$$\mathit{WorkingTimes}(u,t;n,y) \leftarrow \mathit{WorkingSchedules}(u,d;n,y), \mathit{DayTime}(d,t).$$

$$\mathit{PatientUnit}(u,t;p) \leftarrow \mathit{PatientWard}(w,d;p), \mathit{DayTime}(d,t), \mathit{UnitWard}(u,w). \quad (9)$$

3. The set of *quality predicates*, $\mathcal{P}$, with their definitions in, say non-recursive Datalog[8] (possibly with negation, not), in terms of categorical predicates in $\mathcal{R}$ and built-in predicates. They may have partial or full extensions in the contextual instance I (that includes $\mathcal{D}_{\mathcal{M}}$). A quality predicate reflects an application dependent specific quality concern.

Now, $\mathit{TakenByNurse}$ and $\mathit{TakenWithTherm}$ are quality predicates with definitions on top of $\mathcal{M}$, addressing quality concerns about the nurses and the thermometers:

$$\mathit{TakenByNurse}(t,p,n,y) \leftarrow \mathit{WorkingTimes}(u,t;n,y), \mathit{PatientUnit}(u,t;p). \quad (10)$$

$$\mathit{TakenWithTherm}(t,p,b) \leftarrow \mathit{PatientUnit}(u,t;p), u = \texttt{Standard}, b = \texttt{B1}. \quad (11)$$

Furthermore, and not strictly inside context $\mathfrak{C}$, there are predicates $R_1^q, ..., R_n^q \in \mathcal{S}^q$, the *quality versions* of $R_1, ..., R_n \in \mathcal{S}$. They are defined through *quality data extraction rules* written in non-recursive Datalog, in terms of nickname predicates (in $\mathcal{S}'$), categorical predicates (in $\mathcal{R}$), and the quality predicates (in $\mathcal{P}$), and built-in predicates. Their definitions (the α_i^q in Fig. 4) impose conditions corresponding to user's data quality profiles, and their extensions form the quality data (instance).

The quality version of $\mathit{Measurements}$ is $\mathit{Measurement}^q \in \mathcal{S}^q$, with the following definition, which captures the intended, clean contents of the former:

$$\mathit{Measurement}^q(t,p,v) \leftarrow \mathit{Measurement}'(t,p,v), \mathit{TakenByNurse}(t,p,n,y), \quad (12)$$
$$\mathit{TakenWithTherm}(t,p,b), b = \texttt{B1}, y = \texttt{certified}.$$

[7] A *tgd* may support multidimensional navigation and in multiple directions.

[8] Actually, more general rules could be used if they do not increase the complexity of query answering with the MD ontology.

Quality data can be obtained from the interaction between the original source D and the context $\mathfrak{C}$, in particular using the MD ontology $\mathcal{M}$. For that, queries have to be posed to the context, in terms of predicates S^q, the quality versions of those of D. A query could be as direct as asking, e.g. about the contents of predicate $Measurement^q$ above, or a conjunctive query involving predicates S^q.

A naive user —not familiar with the exact interaction with the context— who expects to obtain quality data from D will express a query $\mathcal{Q}$ in terms of the original schema $\mathcal{S}$. However, the information system will rewrite the query into $\mathcal{Q}^q$, in terms of the predicates in $\mathcal{S}^q$. Consequently, the *quality answers* to $\mathcal{Q}$, are defined as those that are *certain* through the context:

Definition 2. For D an instance for schema $\mathcal{S}$, $\mathfrak{C}$ the context containing MD ontology $\mathcal{M}$, and definitions $\Sigma^{\mathcal{P}}, \Sigma^q$ of quality and quality version predicates, resp., the set of *clean answers* to a conjunctive query $\mathcal{Q}(\bar{x})$ on schema $\mathcal{S}$ is:

$$QAns_D^{\mathfrak{C}}(\mathcal{Q}) \ = \ \{\bar{c} \mid D \cup \mathcal{M} \cup \Sigma^{\mathcal{P}} \cup \Sigma^q \ \models \ \mathcal{Q}^q[\bar{c}]\}. \qquad \blacksquare$$

For example, this is the initial query asking for (quality) values for Tom Waits' temperature: $\mathcal{Q}(t,v):\ \ Measurements(t, \texttt{Tom Waits}, v) \wedge \texttt{Sep5-11:45} \leq t \leq \texttt{Sep5-12:15}$, which, in order to be answered, has to be first rewritten into: $\mathcal{Q}^q(t,v):$ $Measurements^q(t, \texttt{Tom Waits}, v) \wedge \texttt{Sep5-11:45} \leq t \leq \texttt{Sep5-12:15}$.

To answer this query, first (12) can be used, obtaining a contextual query:

$$\begin{aligned}\mathcal{Q}^{\mathfrak{C}}(t,v){:}\ \ & Measurement'(t,p,v) \wedge TakenByNurse(t,p,n,\texttt{certified}) \ \wedge \\ & TakenWithTherm(t,p,\texttt{B1}) \wedge p = \texttt{Tom Waits} \ \wedge \\ & \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.\end{aligned}$$

This query will in turn, use the contents for $Measurement'$ coming from D, and the quality predicate definitions (10) and (11), eventually leading to a conjunctive query expressed in terms of $Measurement'$ and MD predicates only, namely:

$$\begin{aligned}\mathcal{Q}^{\mathcal{M}}(t,v){:}\ \ & Measurement'(t,p,v) \wedge WorkingTimes(u,t;n,y) \ \wedge \\ & PatientUnit(u,t;p) \wedge \ u{=}\texttt{Standard} \ \wedge \ y{=}\texttt{certified} \ \wedge \\ & p = \texttt{Tom Waits} \ \wedge \ \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.\end{aligned}$$

At this point, QA from a weakly-sticky ontology has to be performed. We know that this can be done in polynomial time in data. However, there is still a need for practical QA algorithms. Doing this goes beyond the scope of this paper. In [30] we describe some ideas on the development and optimization of such an algorithm.

7 Conclusions

Contexts, in particular, the multidimensional ones introduced in this work, allow us to specify data quality conditions, and to retrieve quality data. This is done by first mapping a data source, possibly with dirty data, into the context. The quality data can be materialized (possibly generating more than one intended clean instance) or be virtually defined. In both cases, it can be retrieved via queries. This latter idea of cleaning data on-the-fly is reminiscent of *consistent query answering* [5]. The main and important difference is that, instead of having (possibly violated) integrity constraints, with contexts we have a much more complex semantic framework for the definition of "repairs" (intended clean instances in our case) and consistent answers (the certain clean answers here).

There is still much to do in terms of development and optimization of practical query answering algorithms for weakly-sticky ontologies. Some first steps are reported in [30]. Implementation and experiments are matter of future work.

Several extensions of the current work have been or are being investigated. Those extensions can be found in the extended version of this paper [31, appendix **??**]. Some of them are as follows:

1. Uncertain downward-navigation when *tgds* allow existentials on categorical attributes. A parent in a category may have multiple children in the next lower category. Under the assumption of complete categorical data, we know it is one of them, but not which one.

2. Our MD ontologies fully capture the taxonomy-based data model [27, 28] and its taxonomy relational algebra (TRA) for query answering. Our appraoch goes beyond [28] in the sense that, first, our categorical relations, by having non-categorical attributes, generalize t-relations. Secondly, the dimensional rules in our MD ontologies capture the TRA, and offer existential variables for handling incomplete data. Finally, we also include and support ontological constraints, such as NCs and *egds* for restricting dimension navigation.

3. The negative constraints (and *egds*, mainly in the separable case) can and are checked on the result of the chase. We think a more natural and practical approach would be to integrate constraint checking with data generation, restricting the latter process. This would amount to compiling constraints into *tgds*, which might lead to the use of negation in *tgd* bodies [13, 20].

4. We may relax the assumption on complete categorical data. This brings many new issues and problems that require investigation; from query answering to the maintenance of *structural semantic constraints*, such as strictness and homogeneity, on the HM model and our extension of it.

References

[1] A. Artale, D. Calvanese, R. Kontchakov and M. Zakharyaschev. The DL-Lite Family and Relations. *J. Artif. Intell. Res.*, 36, 2009, pp. 1-69.

[2] M. Alviano, W. Faber, N. Leone and M. Manna. Disjunctive Datalog with Existential Quantifiers: Semantics, Decidability, and Complexity Issues. *TPLP*, 2012, 12:701-718.

[3] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[4] L. Bertossi, F. Rizzolo and J. Lei. Data Quality is Context Dependent. *Proc. VLDB WS BIRTE'10*, Springer LNBIP 48, 2011, pp. 52-67.

[5] L. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool, 2011.

[6] C. Bolchini, E. Quintarelli and L. Tanca. CARVE: Context-Aware Automatic View Definition over Relational Databases. *Information Systems*, 2013, 38:45-67.

[7] A. Cali, D. Lembo and R. Rosati. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. *Proc. PODS*, 2003, pp. 260-271.

[8] A. Cali, G. Gottlob and T. Lukasiewicz. Datalog$^{\pm}$: A Unified Approach to Ontologies and Integrity Constraints. *Proc. ICDT*, 2009, pp. 14-30.

[9] A. Cali, G. Gottlob, T. Lukasiewicz, B. Marnette and A. Pieris. Datalog$^{\pm}$: A Family of Logical Knowledge Representation and Query Languages for New Applications. *Proc. LICS*, 2010, pp. 228-242.
[10] A. Cali, G. Gottlob and A. Pieris. Query Answering under Non-Guarded Rules in Datalog+/-. *Proc. RR*, 2010, pp. 1-17.
[11] A. Cali, G. Gottlob and A. Pieris. Ontological Query Answering under Expressive Entity-Relationship Schemata. *Information Systems*, 2012, 37(4):320-335.
[12] A. Cali, G. Gottlob and A. Pieris. Towards More Expressive Ontology Languages: The Query Answering Problem. *Artificial Intelligence*, 2012, 193:87-128.
[13] A. Cali, G. Gottlob and Th. Lukasiewicz. A General Datalog-Based Framework for Tractable Query Answering over Ontologies. *Journal of Web Semantics*, 2012, 14:57-83.
[14] A. Cali, M. Console, and R. Frosini. On Separability of Ontological Constraints. *Proc. AMW*, 2012, pp. 48-61.
[15] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi and D. F. Savo. The MASTRO System for Ontology-Based Data Access. *Semantic Web*. 2011, 2(1):43-53.
[16] E. Franconi and U. Sattler. A Data Warehouse Conceptual Data Model for Multidimensional Aggregation. *Proc. DMDW*, CEUR Proceedings, Vol. 19, 1999.
[17] E. Franconi and U. Sattler. A Data Warehouse Conceptual Data Model For Multidimensional Aggregation: A Preliminary Report. *AI*IA Notizie*, 1999, 1:9-21.
[18] G. Gottlob, G. Orsi and A. Pieris. Query Rewriting and Optimization for Ontological Databases. *ACM Trans. Database Syst.*, 2014, 39(3):25.
[19] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science*, 2005, 336:89-124.
[20] A. Hernich, C. Kupke, T. Lukasiewicz and G. Gottlob Well-Founded Semantics for Extended Datalog and Ontological Reasoning. *Proc. PODS*, 2013, pp. 225-236.
[21] C. Hurtado and A. Mendelzon. OLAP Dimension Constraints. *Proc. PODS*, 2002, pp. 169-179.
[22] T. Imielinski and W. Lipski Incomplete Information in Relational Databases. *Journal of the ACM*, 1984, 31(4):761-791.
[23] L. Jiang, A. Borgida and J. Mylopoulos. Towards a Compositional Semantic Account of Data Quality Attributes. *Proc. ER*, 2008, pp. 55-68.
[24] A. Maleki, L. Bertossi and F. Rizzolo. Multidimensional Contexts for Data Quality Assessment. *Proc. AMW*, 2012, CEUR Proceedings, Vol. 866, pp. 196-209.
[25] M. Lenzerini. Data Integration: A Theoretical Perspective. *Proc. PODS*, 2002, pp. 233-246.
[26] N. Leone, M. Manna, G. Terracina, and P. Veltri. Efficiently Computable Datalog$^{\exists}$ Programs. *Proc. KR*, 2012, pp. 1323.
[27] D. Martinenghi and R. Torlone. Querying Context-Aware Databases. *Proc. FQAS*, 2009, pp. 76-87.
[28] D. Martinenghi and R. Torlone. Taxonomy-Based Relaxation of Query Answering in Relational Databases. *The VLDB Journal*, 2014, 23(5):747-769.
[29] M. Milani, L. Bertossi and S. Ariyan. Extending Contexts with Ontologies for Multidimensional Data Quality Assessment. *Proc. ICDEW (DESWeb)*, 2014, pp. 242 - 247.
[30] M. Milani and L. Bertossi. Tractable Query Answering and Optimization for Extensions of Weakly-Sticky Datalog$\pm$. Submitted, under review, 2015.
[31] M. Milani and L. Bertossi. Ontology-Based Multidimensional Contexts with Applications to Quality Data Specification and Extraction. Extended version of this paper. http://people.scs.carleton.ca/~bertossi/papers/obmcExt.pdf
[32] R. Reiter. Towards a Logical Reconstruction of Relational Database Theory. In *On Conceptual Modelling*, M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds.), Springer, 1984, pp. 191-233.