Multidimensional Ontologies for Contextual Quality Data Specification and Extraction

Mostafa Milani[⋆]

Carleton University, School of Computer Science
Ottawa, Canada
mmilani@scs.carleton.ca

Abstract. Data quality is context-dependent. That is, the quality of data cannot be assessed without contextual knowledge about the production or the use of data. 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 a context model for the assessment of the quality of a database instance that was proposed in a previous work [1]. In that framework, the context takes the form of a possibly virtual database or data integration system into which a database instance under quality assessment is mapped, for additional analysis and processing, enabling quality data extraction. In this work we extend contexts with dimensions, and by doing so, we make possible a multidimensional data quality assessment. Multidimensional contexts are represented as ontologies written in Datalog$\pm$. We use this language for representing *dimensional constraints*, and *dimensional rules*, and also for doing *query answering* based on dimensional navigation, which becomes an important auxiliary activity in the assessment of data. We show ideas and mechanisms by means of examples.

1 Introduction

In a previous work [1], a model of context for data quality assessment was proposed. In that work, the assessment of a database D is performed by *putting D in context* or, more precisely, by mapping it into a context $\mathcal{C}$, which is represented as another database, or as a database schema with partial information, or, more generally, as a virtual data integration system [8].

The quality of data in D is determined through additional processing of the data within the context. Data processing in the context leads to possible several quality versions of D, forming a class $\mathcal{D}^q$ of intended, clean versions of D. The quality of D is measured in terms of how much D departs from (its quality versions in) $\mathcal{D}^q$: $dist(D, \mathcal{D}^q)$. 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 $\mathcal{Q}$, from each of the instances in $\mathcal{D}^q$ [7]. The certain answers become what we call the *clean answers* to $\mathcal{Q}$ from D [1].

Problem Statement and Relevance: An important contextual element was not considered in [1]: *dimensions*. 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

[⋆] Supervisor: Leopoldo Bertossi (bertossi@scs.carleton.ca) data model [6], whose inception was mainly motivated by data warehouses (DWH) and OLAP applications.

In Example 1, we illustrate the intuition behind multidimensional context for quality specification and extraction. We assume, according to the HM model, that a dimension consists of a finite set of categories related to each other by a partial order.

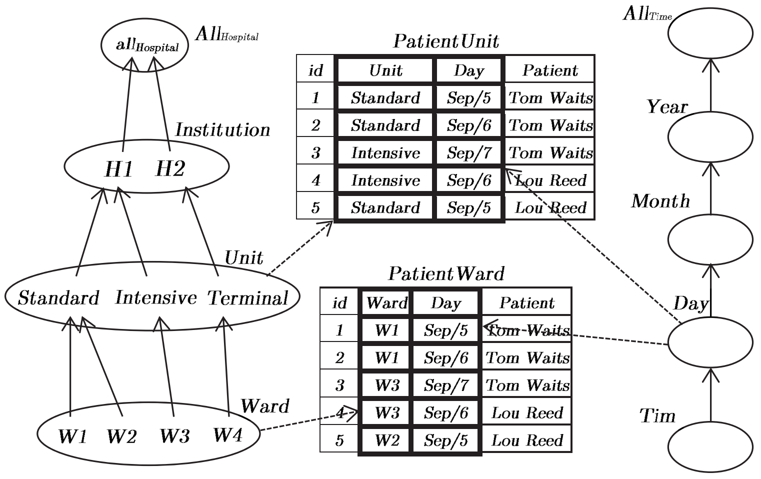
Example 1. The relational table *Measurements* (Table 1) shows body temperatures of patients in an institution. A doctor wants to know *"The body temperatures of Tom Waits for September 5 taken around noon with a thermometer of brand B1"* (as he expected). It is possible that a nurse, unaware of this 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 table *PatientWard* stores the wards of the patients in an institution on different days. The relation is linked to the Hospital and Time dimensions (Fig. 1, middle, bottom) meaning that the ward and day attributes of the relation take values from the members of the *Ward* and the *Day* categories in the Hospital and Time dimensions resp.

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

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 the table *PatientUnit* (Fig. 1, middle, top), which is linked to the *Unit* category, and whose data are (at least partially) generated from *PatientWard* by moving data upward through dimension Hospital (Fig. 1, left), from category *Ward* to category *Unit*.



PatientUnit

id	Unit	Day	Patient
1	Standard	Sep/5	Tom Waits
2	Standard	Sep/6	Tom Waits
3	Intensive	Sep/7	Tom Waits
4	Intensive	Sep/6	Lou Reed
5	Standard	Sep/5	Lou Reed

PatientWard

id	Ward	Day	Patient
1	W1	Sep/5	Tom Waits
2	W1	Sep/6	Tom Waits
3	W3	Sep/7	Tom Waits
4	W3	Sep/6	Lou Reed
5	W2	Sep/5	Lou Reed

Fig. 1. An extended multidimensional model

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 $Measurements^q$.

Elaborating on this example, there could be a constraint involving the *Unit* category in the Hospital dimension: *"No patient in intensive care unit at any time after August /2005"*. This constraint could be applied on the process of moving data upward from *PatientWard* to *PatientUnit*. ■

Example 1 illustrates the necessity of a formal data model to represent multidimensional context (e.g. including relations linked to dimensions, constraints and rules on these relations).

Proposed Solutions: In this work, 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, i.e. *PatientWard* and *PatientUnit* (think of generalized fact tables as found in data warehouses). It also include *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*.

Categorical relations may be incomplete, and new data can be generated for them, which will be enabled through rules (*tgds*) of a Datalog$\pm$ dimensional ontology. The previous example shows data generation via upward navigation while *downward navigation* may also be useful (cf. [13]). Our approach to multidimensional contexts will support both.

We propose an ontological representation in Datalog$^\pm$ [3, 4] of the extended HM model, and also mechanisms for data quality assessment based on query answering (QA) from the ontology via dimensional navigation. 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. A *multidimensional context* —corresponding to the formalization of the extension of HM— becomes a Datalog$\pm$ ontology, $\mathcal{M}$, that belongs to an interesting syntactic classes of programs (called weakly-sticky [5]), for which some results are known. This allows us to give a semantics to our ontologies, and apply some established and new algorithms for QA.

The proposed multidimensional context forms a weakly-sticky ontology. This implies that quality query answering and quality data extraction require an algorithm for QA from such ontologies and its optimization and implementation. The weakly-sticky class of ontologies is a well-established member of the family of Datalog$^\pm$ ontologies with polynomial time data complexity for QA [5]. However, a practical QA algorithm for this member of Datalog$^\pm$ is still missing in the literature. Here, we propose a practical QA algorithm that runs in polynomial time and it is applicable for some members of the Datalog$^\pm$ family, including weakly-sticky ontologies. We also study the magic-sets optimization techniques and show that they are applicable on the QA algorithm.

2 Work Done So Far

We enhance the approach to data quality specification and extraction in [1] (Figure 2), by adding dimensions to contexts. In this case, the context contains a generic multidimensional ontology, the shaded $\mathcal{M}$ in Fig. 3, aka. "core ontology" (described next). This ontology can be extended, within the context, with additional rules and constraints that depend on specific data quality concerns (cf. Section 2.3).

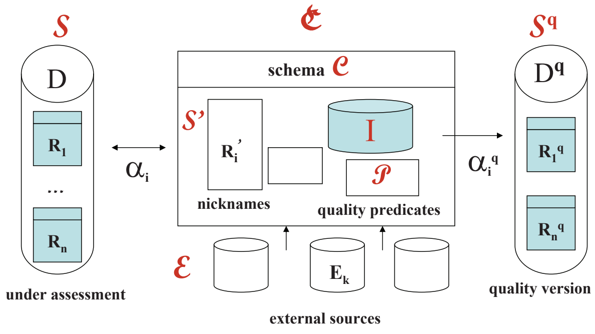


Fig. 2. A context for data quality assessment

2.1 The Extended MD Model and 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$\pm$ 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 2. Categorical relation $PatientWard(Ward,Day;Patient)$ in Fig. 1 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. 3. 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 (NC)*:[1] ($R \in \mathcal{R}$, $K \in \mathcal{K}$; $\bar{e}, \bar{a}$ are categorical, non-categorical, resp.; $e \in \bar{e}$)

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

[1] 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.

(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))

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

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

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

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

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

With rule (4) (an example is (7) below), the possibility of doing dimensional navigation is captured by joins between categorical predicates, e.g. $R_i(\bar{\boldsymbol{e}}_i; \bar{a}_i), ..., R_j(\bar{\boldsymbol{e}}_j; \bar{a}_j)$ in the body, and parent-child predicates, e.g. $D_n(e_n, e_n'), ..., D_m(e_m, e_m')$.

Rule (4) allows navigation in both upward and downward directions. The *direction of navigation* is determined by the level of categorical attributes that participate in the join in the body. Assuming the join is between $R_i(\bar{e}_i; \bar{a}_i)$ and $D_n(e_n, e_n')$, upward navigation is enabled when $e_n' \in \bar{e}_i$ (i.e. e_n' appears in $R_i(\bar{e}_i; \bar{a}_i)$) and $e_n \in \bar{e}_k$ (i.e e_n appears in the head). On the other hand, if e_n occurs in R_i and e_n' occurs in R_k, then downward navigation is enabled, from e_n to e_n'.

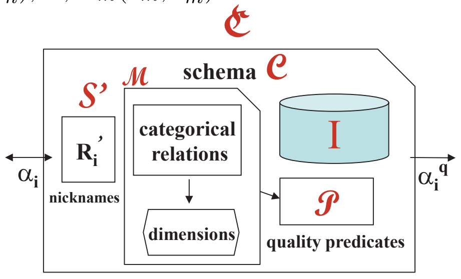


Fig. 3. A multidimensional context

Example 3. (example 2 cont.) The categorical attribute *Unit* in categorical relation *PatientUnit* takes values from the *Unit* category. We use a constraint of the form (1):
$\bot \ \leftarrow \ \mathit{PatientUnit}(\boldsymbol{u}, \boldsymbol{d}; p), \neg \mathit{Unit}(u).$ (5)

For the constraint in Example 1 requiring *"No patient was in intensive care unit during the time after August 2005"*, we use a dimensional constraint of the form (3):

$$\bot \ \leftarrow \ [\mathit{PatientWard}(\boldsymbol{w}, \boldsymbol{d}; p), \mathit{UnitWard}(\texttt{Intensive}, w), \quad (6)$$
$$\mathit{MonthDay}(\texttt{August/2005}, d)].$$

The following dimensional rules of the form (4) capture how data in *PatientWard* and *WorkingSchedules* generate data for *PatientUnit* and *Shifts*, resp.:

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

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

In (7), dimension navigation is enabled by the join between *PatientWard* and *UnitWard*. The rule generates data for *PatientUnit* (at a the higher level of *Unit*) from *PatientWard* (at the lower level of *Ward*) via upward navigation.

Rule (8) captures a guideline that 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"*. The rule is expressed using two additional categorical relations, *WorkingSchedules* and *Shifts*, that store schedules of nurses in units and shifts of nurses in wards, resp. Downward navigation is performed by generating data for *Shifts* (at the level of *Ward*) from *WorkingSchedules* (at the level of *Unit*). In this case, the schemas of the two categorical relations do not match. So, the existential variable z represents missing data for the *shift* attribute. ■

2.2 Properties of MD Datalog$^\pm$ Ontologies

Here, we first establish the membership of our MD ontologies, $\mathcal{M}$ (cf. Section 2.1) of a class of the Datalog$\pm$ 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$\pm$ programs. ■

The proof and a review of *weakly-sticky* Datalog$\pm$ [5], can be found in [13] (an extended version of [11]). A consequence of this result is that QA from $\Sigma_\mathcal{M}$ is in polynomial-time in data complexity [5].

The complexity stays the same if we add *NCs*, of the forms (1) and (3), because they can be checked through the conjunctive queries in their bodies [5]. However, combining the *egds* in $\epsilon_\mathcal{M}$ with $\Sigma_\mathcal{M}$ could change things, and, in principle, even lead to undecidability of QA [2]. Separability[5] of *egds* and *tgds* is a semantic condition that guarantees complexity of query answering still remains the same even after adding *egds*. In [13] (Proposition 2), we show a syntactic condition for the dimensional *edgs* that implies the separability.

2.3 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 wrt. a database instance D for schema $\mathcal{S}$. Here, Context $\mathfrak{C}$, as shown in Fig. 3, 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, $Measurements' \in \mathcal{S}'$ is a nickname predicate for $Measurements \in \mathcal{S}$, whose initial contents (in D) is under quality assessment.

2. The *core MD ontology*, $\mathcal{M}$. We assume that application dependent guidelines and constraints are all represented as components of $\mathcal{M}$.

In our running example, $PatientUnit$, $PatientWard$, $WorkingSchedules$ and $WorkingTimes$ are categorical relations. $UnitWard$, $DayTime$ are parent-child relations in dimensions, Hospital and Time, resp. The followings are dimensional rules of $\Sigma_\mathcal{M}$:

$$WorkingTimes(u,t;n,y) \leftarrow WorkingSchedules(u,d;n,y),\, DayTime(d,t).$$
$$PatientUnit(u,t;p) \leftarrow PatientWard(w,d;p), DayTime(d,t), UnitWard(u,w). \quad (9)$$

3. The set of *quality predicates*, $\mathcal{P}$, with their definitions in non-recursive Datalog (possibly with negation, not), in terms of categorical predicates in $\mathcal{R}$ and built-in predicates. A quality predicate reflects an application dependent specific quality concern.

Now, $TakenByNurse$ and $TakenWithTherm$ 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. 3) 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:

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

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 $\mathit{Measurement}^q$ above, or a conjunctive query involving predicates S^q.

For example, this is the initial query asking for (quality) values for Tom Waits' temperature: $\mathcal{Q}(t,v) : \mathit{Measurements}(t, \texttt{Tom Waits}, v) \wedge \texttt{Sep5-11:45} \leq t \leq \texttt{Sep5-12:15}$, which, in order to be answered, has to be first rewritten into, $\mathcal{Q}^q(t,v) : \mathit{Measurements}^q (t, \texttt{Tom Waits}, v) \wedge \texttt{Sep5-11:45} \leq t \leq \texttt{Sep5-12:15}$.
To answer this query, first (12) can be used, obtaining a contextual query: $\mathcal{Q}^{\mathfrak{C}}(t,v) : \mathit{Measurement}'(t,p,v) \wedge \mathit{TakenByNurse}(t,p,n,\texttt{certified}) \wedge \mathit{TakenWithTherm}(t, p, \texttt{B1}) \wedge p = \texttt{Tom Waits} \wedge \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}$.

This query will in turn, use the contents for $\mathit{Measurement}'$ coming from D, and the quality predicate definitions (10) and (11), eventually leading to a conjunctive query expressed in terms of $\mathit{Measurement}'$ and MD predicates only, namely: $\mathcal{Q}^{\mathcal{M}}(t,v) : \mathit{Measurement}'(t,p,v) \wedge \mathit{WorkingTimes}(u,t;n,y) \wedge \mathit{PatientUnit}\ (u,t;p) \wedge u = \texttt{Standard} \wedge y = \texttt{certified} \wedge p = \texttt{Tom Waits} \wedge \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}$.

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 Section 2.4, we describe some ideas from [12] on the development and optimization of such an algorithm.

2.4 Query Answering on the MD Ontology

We proposed a conjunctive query answering algorithm for weakly sticky programs to be also applied to our MD ontology. The algorithm is based on the concepts of *parsimonious chase* (*pChase*) and *freezing nulls*, as used for QA with *shy Datalog*, a fragment of *Datalog*$^\exists$ [9].

At a *pChase* step, a new atom is added only if a homomorphic atom is not already in the chase. Freezing a null is promoting it to a constant (and keeping it as such in subsequent chase steps). So, it cannot take (other) values under homomorphisms, which may create new *pChase* steps. Resumption of the *pChase* means freezing *all* nulls, and continuing *pChase* until no more *pChase* steps are applicable.

Query answering with shy programs has a first phase where the *pChase* runs until termination. In a second phase, the *pChase* iteratively resumes for a number of times that depends on the number of distinct $\exists$-variables in the query. This second phase is required to properly deal with joins in the query. Our algorithm for weakly sticky programs is similar, it has the same two phases, but a *pChase* step is modified: after every application of a *pChase* step that generates nulls, the latter that appear in positions with finite ranks are immediately frozen. The algorithm runs in polynomial-time in data.

3 Work Still to Be Done

We are investigating several extensions of the current work. 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) Using the MD ontologies to fully capture *the taxonomy-based data model* [10], (3) 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] L. Bertossi, F. Rizzolo and J. Lei. Data Quality is Context Dependent. *Proc. VLDB WS (BIRTE'10)*, Springer LNBIP 48, 2011, pp. 52-67.

[2] 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.

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

[4] 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.

[5] A. Cali, G. Gottlob and A. Pieris. Towards More Expressive Ontology Languages: The Query Answering Problem. *Artificial Intelligence*, 2012, 193:87-128.

[6] C. Hurtado and A. Mendelzon. OLAP Dimension Constraints. *Proc. PODs*, 2002, pp. 169-179.

[7] T. Imielinski and W. Lipski Incomplete Information in Relational Databases. *Journal of the ACM*, 1984, 31(4):761-791.

[8] M. Lenzerini. Data Integration: A Theoretical Perspective. *Proc. PODS*, 2002, pp. 233-246.

[9] N. Leone, M. Manna, G. Terracina, and P. Veltri. Efficiently Computable Datalog$^{\exists}$ Programs. *Proc. KR*, 2012, pp. 1323.

[10] D. Martinenghi and R. Torlone. Querying Context-Aware Databases. *Proc. FQAS*, 2009, pp. 76-87.

[11] M. Milani, L. Bertossi and S. Ariyan. Extending Contexts with Ontologies for Multidimensional Data Quality Assessment. *Proc. ICDEW (DESWeb)*, 2014, pp. 242 - 247.

[12] M. Milani and L. Bertossi. Tractable Query Answering and Optimization for Extensions of Weakly-Sticky Datalog$\pm$. *Proc. AMW*, 2015.

[13] M. Milani and L. Bertossi. Ontology-Based Multidimensional Contexts with Applications to Quality Data Specification and Extraction. Submitted, Accepted in RuleML, 2015.