The Ontological Multidimensional Data Model
(extended abstract)

Leopoldo Bertossi[⋆] and **Mostafa Milani**[⋆⋆]

Abstract. We briefly present *OMD*, a model of multidimensional data that uses ontologies written in Datalog$^{\pm}$, an extension of the classical declarative language Datalog for relational databases.

We present the *Ontological Multidimensional Data Model* (OMD) as an ontological, Datalog$^{\pm}$-based [3] extension of the Hurtado-Mendelzon (HM) model for multidimensional data [5].

For limitations of space, we will use a running example to illustrate the main elements of an OMD model.

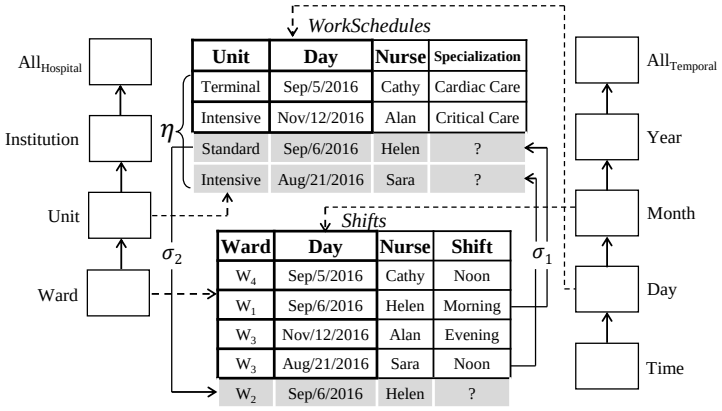
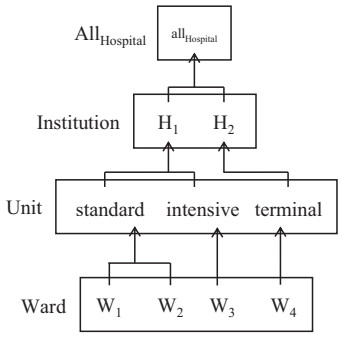


Fig. 1. An OMD model with categorical relations, dimensional rules, and constraints

An OMD model has a *database schema* $\mathcal{R}^{\mathcal{M}} = \mathcal{H} \cup \mathcal{R}^{c}$, where $\mathcal{H}$ is a relational schema with multiple dimensions, with sets $\mathcal{K}$ of unary category predicates, and sets $\mathcal{L}$ of binary, child-parent predicates; and $\mathcal{R}^{c}$ is a set of *categorical predicates*.

Example: Figure 1 shows Hospital and Temporal dimensions. The former's instance is here on the RHS. $\mathcal{K}$ contains predicates $Ward(\cdot)$, $Unit(\cdot)$, $Institution(\cdot)$, etc. Instance $D^{\mathcal{H}}$ gives them extensions, e.g. $Ward = \{\mathsf{W}_1, \mathsf{W}_2, \mathsf{W}_3, \mathsf{W}_4\}$. $\mathcal{L}$ contains, e.g. $WardUnit(\cdot,\cdot)$, with extension: $WardUnit = \{(\mathsf{W}_1, \mathsf{standard}), (\mathsf{W}_2, \mathsf{standard}), (\mathsf{W}_3, \mathsf{intensive}), (\mathsf{W}_4, \mathsf{terminal})\}$. In the middle of Figure 1, *categorical relations* are associated to dimension categories. □



⋆ Carleton Univ., School of Computer Science, Canada. bertossi@scs.carleton.ca
⋆⋆ McMaster Univ., Dept. Computing and Software, Canada. mmilani@mcmaster.ca

Attributes of categorical predicates are either *categorical*, whose values are members of dimension categories, or *non-categorical*, taking values from arbitrary domains. Categorical predicate are represented in the form $R(C_1, \ldots, C_m; N_1, \ldots, N_n)$, with categorical attributes before ";" and non-categorical after.

The extensional data, i.e the instance for the schema $\mathcal{R}^{\mathcal{M}}$, is $I^{\mathcal{M}} = D^{\mathcal{H}} \cup I^c$, where $D^{\mathcal{H}}$ is a complete instance for dimensional subschema $\mathcal{H}$ containing the category and child-parent predicates; and sub-instance I^c contains possibly partial, incomplete extensions for the categorical predicates, i.e. those in $\mathcal{R}^c$.

Schema $\mathcal{R}^{\mathcal{M}}$ comes with basic, application-independent semantic constraints, listed below.

1. Dimensional child-parent predicates must take their values from categories. Accordingly, if child-parent predicate $P \in \mathcal{L}$ is associated to category predicates $K, K' \in \mathcal{K}$, in this order, we introduce inclusion dependencies (IDs) as Datalog$^{\pm}$ *negative constraints* (*ncs*): $P(x, x'), \neg K(x) \rightarrow \bot$, and $P(x, x'), \neg K'(x') \rightarrow \bot$. (The $\bot$ symbol denotes an always false propositional atom.) We do not represent them as Datalog$^{\pm}$'s *tuple-generating dependencies* (*tgds*) $P(x, x') \rightarrow K(x)$, etc., because we reserve *tgds* for possibly incomplete predicates (in their RHSs).

2. Key constraints on dimensional child-parent predicates $P \in \mathcal{K}$, as *equality-generating dependencies* (*egds*): $P(x, x_1), P(x, x_2) \rightarrow x_1 = x_2$.

3. The connections between categorical attributes and the category predicates are specified by means of *ncs*. For categorical predicate R, the *nc* $R(\bar{x}; \bar{y}), \neg K(x) \rightarrow \bot$, where $x \in \bar{x}$ takes values in category K.

Example: The categorical attributes *Unit* and *Day* of categorical predicate *WorkingSchedules*(*Unit*,*Day*;*Nurse*, *Speciality*) in $\mathcal{R}^c$ are connected to the Hospital and Temporal dimensions, resp., as captured by the IDs *WorkingSchedules*[1] $\subseteq$ *Unit*[1], and *WorkingSchedules*[2] $\subseteq$ *Day*[1]. The former is written in Datalog^{+}as *WorkingSchedules*$(u, d; n, t), \neg$*Unit*$(u) \rightarrow \bot$. For the Hospital dimension, one of the IDs for predicate *WardUnit* is *WardUnit*[2] $\subseteq$ *Unit*[1], which is expressed by the *nc*: *WardUnit*$(w, u), \neg$*Unit*$(u) \rightarrow \bot$. The key constraint of *WardUnit* is captured by the *egd*: *WardUnit*(w, u), *WardUnit*$(w, u') \rightarrow u = u'$. □

The OMD model allows us to build *multidimensional ontologies*, $\mathcal{O}^{\mathcal{M}}$. In addition to an instance $I^{\mathcal{M}}$ for a schema $\mathcal{R}^{\mathcal{M}}$, they include the set $\Omega^{\mathcal{M}}$ of *basic constraints* as in **1.-3.** above, a set $\Sigma^{\mathcal{M}}$ of *dimensional rules* (those in **4.** below), and a set $\kappa^{\mathcal{M}}$ of *dimensional constraints* (in **5.** below); all of them application-dependent and expressed in the Datalog^{+} language associated to schema $\mathcal{R}^{\mathcal{M}}$.

4. *Dimensional rules* as Datalog^{+} *tgds*: $R_1(\bar{x}_1; \bar{y}_1), ..., R_n(\bar{x}_n; \bar{y}_n), P_1(x_1, x'_1), ..., P_m(x_m, x'_m) \rightarrow \exists \bar{y}'\ R_k(\bar{x}_k; \bar{y})$. Here, the $R_i(\bar{x}_i; \bar{y}_i))$ are categorical predicates, the P_i are child-parent predicates, $\bar{y}' \subseteq \bar{y}$, $\bar{x}_k \subseteq \bar{x}_1 \cup ... \cup \bar{x}_n \cup \{x_1, ..., x_m, x'_1, ..., x'_m\}$, $\bar{y} \smallsetminus \bar{y}' \subseteq \bar{y}_1 \cup ... \cup \bar{y}_n$; repeated variables in bodies (join variables) appear only categorical positions in categorical relations and in child-parent predicates. Existential variables appear only in non-categorical attributes.

5. *Dimensional constraints*, as *egds* or *ncs*: $R_1(\bar{x}_1; \bar{y}_1), ..., R_n(\bar{x}_n; \bar{y}_n), P_1(x_1, x'_1), ..., P_m(x_m, x'_m) \rightarrow z = z'$, and $R_1(\bar{x}_1; \bar{y}_1), ..., R_n(\bar{x}_n; \bar{y}_n), P_1(x_1, x'_1), ..., P_m(x_m, x'_m) \rightarrow \bot$. Here, $R_i \in \mathcal{R}^c$, $P_j \in \mathcal{L}$, and $z, z' \in \bigcup \bar{x}_i \cup \bigcup \bar{y}_j$. Some

of the lists in the bodies may be empty, i.e. $n = 0$ or $m = 0$, which allows to represent also classical constraints on categorical relations, e.g. keys or FDs.

Example: The left-hand-side of Figure 1 shows *dimensional constraint* η on categorical relation *WorkingSchedules*, which is linked to the Temporal dimension via the *Day* category. It says: *"No personnel was working in the Intensive care unit in January"*, i.e. η: $\mathit{WorkingSchedules}(\mathsf{intensive}, d; n, s), \mathit{DayMonth}(d, \mathsf{jan}) \rightarrow \bot$.

Dimensional *tgd* σ_1 in Figure 1, given by $\mathit{Shifts}(w, d; n, s), \mathit{WardUnit}(w, u) \rightarrow \exists t\ \mathit{WorkingSchedules}(u, d; n, t)$, says that *"If a nurse has shifts in a ward on a specific day, he/she has a working schedule in the unit of that ward on the same day"*. The use of σ_1 generates, from the *Shifts* relation, new tuples for relation *WorkingSchedules*, with *null values* for the *Specialization* attribute, due to the existential variable. Existential rules like this (and also *egds* and *ncs*) make us depart from classic Datalog, taking us into Datalog$^{\pm}$. Relation *Working Schedules* may be incomplete, and new -possibly virtual- entries can be produced for it, e.g. the shaded ones showing *Helen* and *Sara* working for the *Standard* and *Intensive* units, resp. This is done by *upward-navigation and data propagation* through the dimension hierarchy. Constraint η is expected to be satisfied both by the initial extensional tuples for *WorkingSchedules* and its tuples generated through σ_1, i.e. by its non-shaded tuples and shaded tuples in Figure 1, resp. In this example, η is satisfied.

Notice that *WorkingSchedules* refers to the *Day* attribute of the Temporal dimensions, whereas η involves the *Month* attribute. Then, checking η requires upward-navigation through the Temporal dimension. Also the Hospital dimension is involved in the satisfaction of η: The *tgd* σ_1 may generate new tuples for *WorkingSchedules*, by upward-navigation from *Ward* to *Unit*.

Furthermore, we have an additional *tgd* σ_2 that can be used with *WorkingSchedules* to generate data for categorical relation *Shifts* (the shaded tuple in it is one of them): σ_2 : $\mathit{WorkingSchedules}(u, d; n, t), \mathit{WardUnit}(w, u) \rightarrow \exists s\ \mathit{Shifts}(w, d; n, s)$. It reflects the institutional guideline stating that *"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"*. Accordingly, σ_2 relies on downward-navigation for tuple generation, from the *Unit* category level down to the *Ward* category level.

If we have a categorical relation $\mathit{Therm}(\mathit{Ward}, \mathit{Thertype}; \mathit{Nurse})$, with *Ward* and *Thertype* categorical attributes (the latter for an Instrument dimension), the following is an *egd* saying that *"All thermometers in a unit are of the same type"*: $\mathit{Therm}(w, t; n), \mathit{Therm}(w', t'; n'), \mathit{WardUnit}(w, u), \mathit{WardUnit}(w', u) \rightarrow t = t'$.

Notice that our ontological language allows us to impose a condition at the *Unit* level without having it as an attribute in the categorical relation. The existential variables in dimensional rules, such as t and s as in σ_1 and σ_2, resp., make up for the missing, non-categorical attributes *Speciality* and *Shift* in *WorkingSchedules* and *Shifts*, resp. □

Dimensional *tgds* can be used for *upward-* or *downward-navigation* (or data generation) depending on the joins in the body. A one-step direction is determined by the difference of levels of the dimension categories appearing (as attributes) in the joins. Multi-step navigation, between a category and an ancestor

or descendant category, can be captured through a chain of joins with adjacent child-parent dimensional predicates in the body of a *tgd*, e.g. propagating doctors at the unit level all the way up to the hospital level: $\mathit{WardDoc}(\mathit{ward}; \mathit{na}, \mathit{sp})$, $\mathit{WardUnit}(\mathit{ward}, \mathit{unit})$, $\mathit{UnitInst}(\mathit{unit}, \mathit{ins}) \rightarrow \mathit{HospDoc}(\mathit{ins}; \mathit{na}, \mathit{sp})$.

Example: Rule σ_2 supports downward tuple-generation. When enforcing it on a tuple $\mathit{WorkingSchedules}(u, d; n, t)$, via category member u (for Unit), a tuple for *Shifts* is generated *for each* child w of u in the *Ward* category for which the body of σ_2 is true. For example, chasing σ_2 with the third tuple in *WorkingSchedules* generates two new tuples in *Shifts*: $\mathit{Shifts}(\mathsf{W}_2, \mathsf{sep/6/2016}, \mathsf{helen}, \zeta)$ and $\mathit{Shifts}(\mathsf{W}_1, \mathsf{sep/6/2016}, \mathsf{helen}, \zeta')$, with fresh nulls, ζ and ζ'. The latter tuple is not shown in Figure 1 (it is dominated by the third tuple, $\mathit{Shifts}(\mathsf{W}_1, \mathsf{sep/6/2016}, \mathsf{helen}, \mathsf{morning})$, in *Shifts*). With the old and new tuples we obtain the answers to the query about *Helen*'s wards on *Sep/6/2016*: $\mathcal{Q}'(w)$: $\exists s\ \mathit{Shifts}(w, \mathsf{sep/6/2016}, \mathsf{helen}, s)$. They are W_1 and W_2.

In contrast, the join between *Shifts* and *WardUnit* in σ_1 enables upward-navigation; and the generation of only one tuple for *WorkingSchedules* from each tuple in *Shifts*, because each *Ward* member has at most one *Unit* parent. □

We can see that the OMD data model is an ontological model that goes far beyond classical multidimensional data models. For example, the HM model [5], which is subsumed by OMD, does not include general *tgds*, *egds*, or *ncs*. Starting from our relational reconstruction of the HM model, all these elements, plus the data and queries, are seamlessly integrated into a uniform logico-relational framework. OMD supports general, possibly incomplete categorical relations, and not only complete "fact tables" linked to base (or bottom) categories.

Furthermore, the constraints considered in the HM model are specific for the dimensional structure of data, most prominently, to guarantee summarizability (i.e. correct aggregation, avoiding double-counting). Specifically, we find constraints enforcing *strictness* and *homogeneity* [5]. The former requires that every category elements rolls-up to a single element in a parent category, which in OMD can be expressed by *egds*. The latter requires that category elements have parent elements in parent categories, which in OMD can be expressed by *tgds*. (Cf. [10, sec. 4.3] for more details.)

The OMD model enables *ontology-based data access* (OBDA) [6] and allows for the tight integration of conceptual models (e.g. an ER model expressed in logical terms) and the relational model of data, while representing and using dimensional structures and data. Cf. [7, 2] for applications of the OMD model to quality data specification and extraction.

The ontologies of the OMD model have good computational properties [2, 7]. Actually, they belong to the class of *weakly-sticky* Datalog$^{\pm}$ programs [4], for which conjunctive query answering (CQA) can be done in polynomial time in data. Algorithms for CQA have been proposed [8, 9], so as optimizations thereof [8] with *magic-sets* techniques [1].

Acknowledgements: Research supported by NSERC Discovery Grant #06148.

References

[1] M. Alviano, N. Leone, M. Manna, G. Terracina and P. Veltri. Magic-Sets for Datalog with Existential Quantifiers. *Proc. Datalog 2.0*, Springer LNCS 7494, 2012, pp. 31-43.

[2] Bertossi, L. and Milani, M. Ontological Multidimensional Data Models and Contextual Data Quality. Journal submission, 2017. Posted as Corr Arxiv Paper cs.DB/1704.00115.

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

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

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

[6] M. Lenzerini. Ontology-Based Data Management. Proc. AMW 2012, CEUR Proceedings, Vol. 866, pp. 12-15.

[7] Milani, M. and Bertossi, L. Ontology-Based Multidimensional Contexts with Applications to Quality Data Specification and Extraction. *Proc. RuleML*, Springer LNCS 9202, 2015, pp. 277-293.

[8] Milani, M. and Bertossi, L. Extending Weakly-Sticky Datalog$^{\pm}$: Query-Answering Tractability and Optimizations. *Proc. RR*, Springer LNCS 9898, 2016, pp. 128-143.

[9] Milani, M., Bertossi, L. and Calì, A. A Hybrid Approach to Query Answering under Expressive Datalog$^{\pm}$. *Proc. RR*, Springer LNCS 9898, 2016, pp. 144-158.

[10] Milani. M. *Multidimensional Ontologies for Contextual Quality Data Specification and Extraction*. PhD Thesis, Carleton University, January 2017. http://people.scs.carleton.ca/~bertossi/papers/mostafaFinal.pdf