Contexts and Multidimensional Data Quality Assessment and Cleaning

Leopoldo Bertossi
Carleton University
Ottawa, Canada
(Faculty Fellow IBM CAS)

Data Quality: An Activity in Flux

Data quality assessment (DQA) and data cleaning (DC) as activities have been mostly:

- *ad-hoc*
- rigid
- vertical
- application-dependent
- mechanism-based

There has been a lack of fundamental research in DQA and DC

Natural questions:

- Any general principles underlying DQA and DC?
- General methodologies?
- Parameterized generic methods to be instantiated on different scenarios?
- Semantics behind solutions (if any)?
- Declarative solutions with a clear semantics?
- Scope of applicability of different solutions?

Things are starting to change ...

Semantic Constraints for Data Quality

Recently, DQ constraints have been proposed and investigated

They provide generic languages for expressing quality concerns

Suitable for specifying adaptive and generic DQ/C mechanisms

- Conditional dependencies
- Matching dependencies
- ...

Similar to classical integrity constraints (ICs), but with DC in mind

ICs have been around for a long time:

- Capture the application semantics in the data model and DB
- Studied in general and logical terms
- Have wide application in data management
- Large body of applied and fundamental research
- Methodologies for dealing with ICs are quite general and have broad applicability
- Emphasis on:
  - Correspondence between outside reality and the model (the DB)
  - Specify legal DB states and updates
  - Metadata for Inter-operability
  - Semantic query optimization

More recently:

Database *repairing* and *consistent query answering* (CQA)

Newer use for ICs and a contribution to DQA and DC using classical semantic constraints

Characterization of semantically correct data in a possibly inconsistent DB

Obtain quality answers on-the-fly, at query answering time

Paradigm shift: *Constraints on query answers instead of on DB states*

These are all cases of semantic information providing a context for DQA and DC

We can go beyond ...

Contexts and Data Quality

(Join work with Flavio Rizzolo)

A table containing data about the temperatures of patients at a hospital

TempNoon

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

Are these quality data?

If not, is there anything to clean? What?

We do not know ... It depends ...

Actually the table is supposed/expected to contain

“temperature measurements for Tom taken at noon by a certified nurse with an oral thermometer”

We still do not know if these are quality data?

Questions about the quality of this data make sense in a broader setting

The quality of the data depends on “the context”

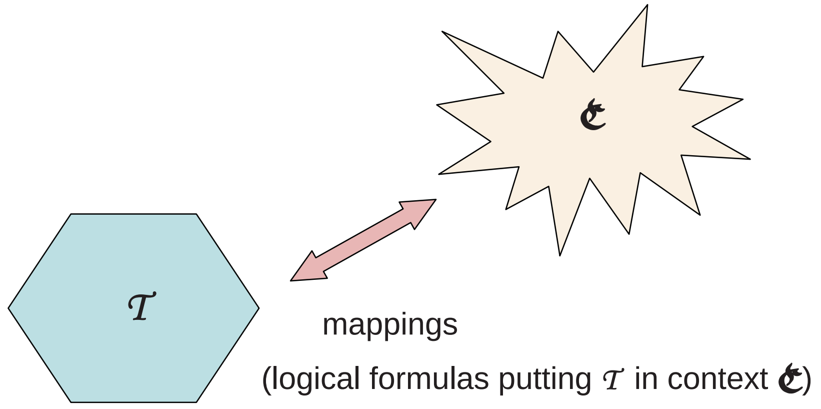
A context that allows us to:

- make sense of the data
- provide semantics to data
- logically and explicitly connect data with other, otherwise implicit, data
- assess data
- support data cleaning

Our approach starts from the fact that quality of data cannot be assessed without contextual knowledge about the production or use of data

Context-based data quality assessment requires a formal model of context

Contexts: A General Vision


$\mathcal{T}$
$\mathfrak{C}$
mappings
(logical formulas putting $\mathcal{T}$ in context $\mathfrak{C}$)

- A logical theory $\mathcal{T}$ is the one that has to be "put in context"

  For example, a relational database can be seen as a theory

- The context is another logical theory, $\mathcal{C}$

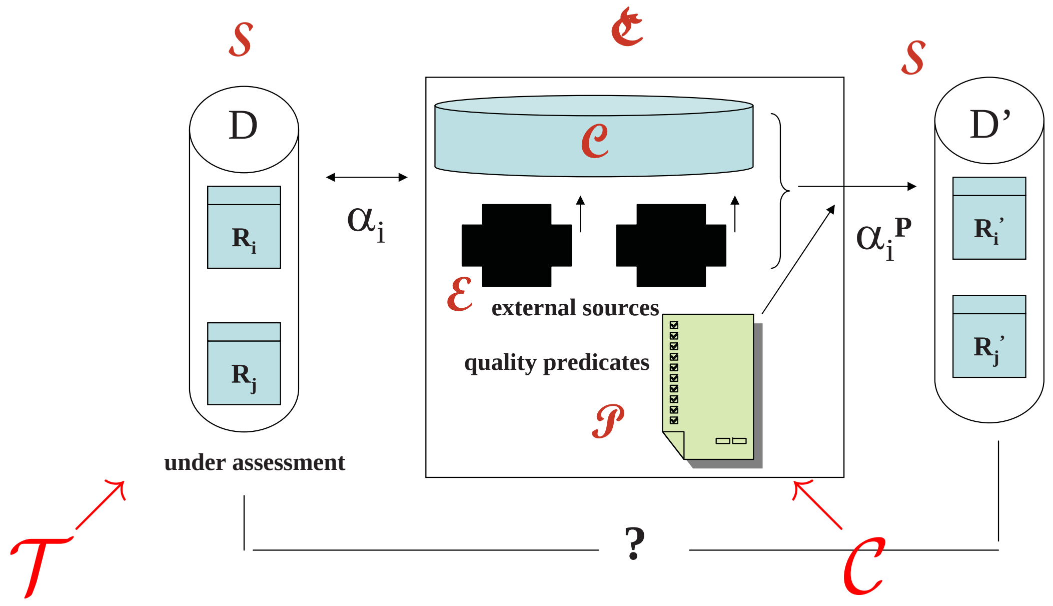
  For example, an ontology, a virtual data integration system

- $\mathcal{T}$ and $\mathcal{C}$ may share some predicate symbols

  Connection between $\mathcal{T}$ and $\mathcal{C}$ is established through (possibly shared) predicates and logical mappings

In particular for applications in data management

In our data quality scenario: (VLDB'10 BIRTE WS, Springer LNBIP 48, 2011)



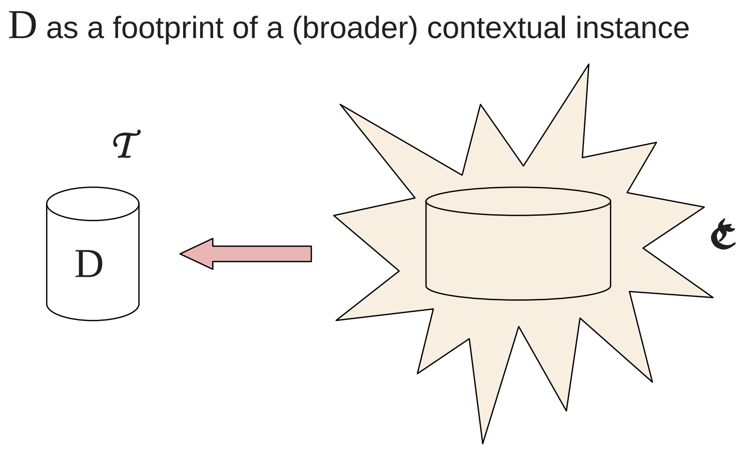
- Database D can be a logical theory

  E.g. Reiter's logical reconstruction of a relational DB

- Context $\mathfrak{C}$ can be a knowledge base, an ontology, another database, ...

More concretely:

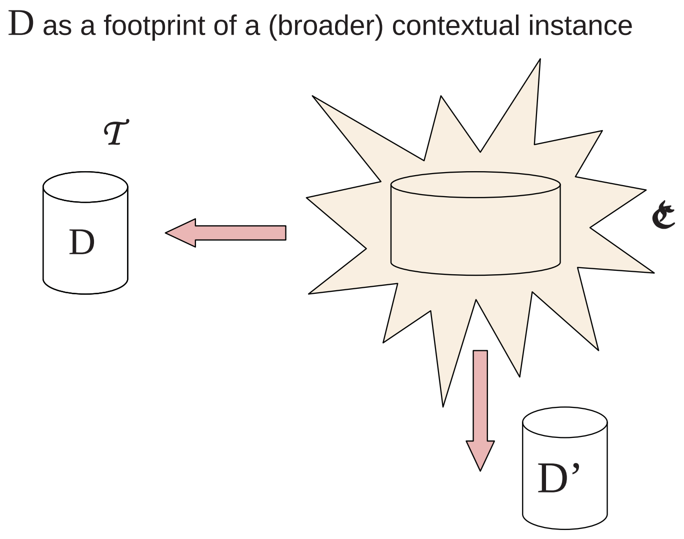
(A) D expected to be a footprint of contextual data



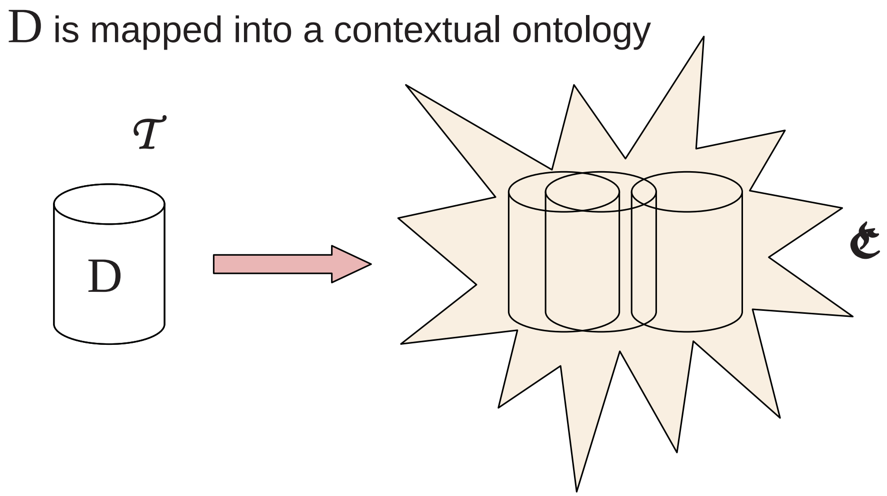
Data in $\mathfrak{C}$ (including D) are analyzed/assessed/cleaned

According to additional data available in or accessible from $\mathfrak{C}$; and quality criteria defined in $\mathfrak{C}$

A new version of D is obtained and can be compared with D for quality assessment



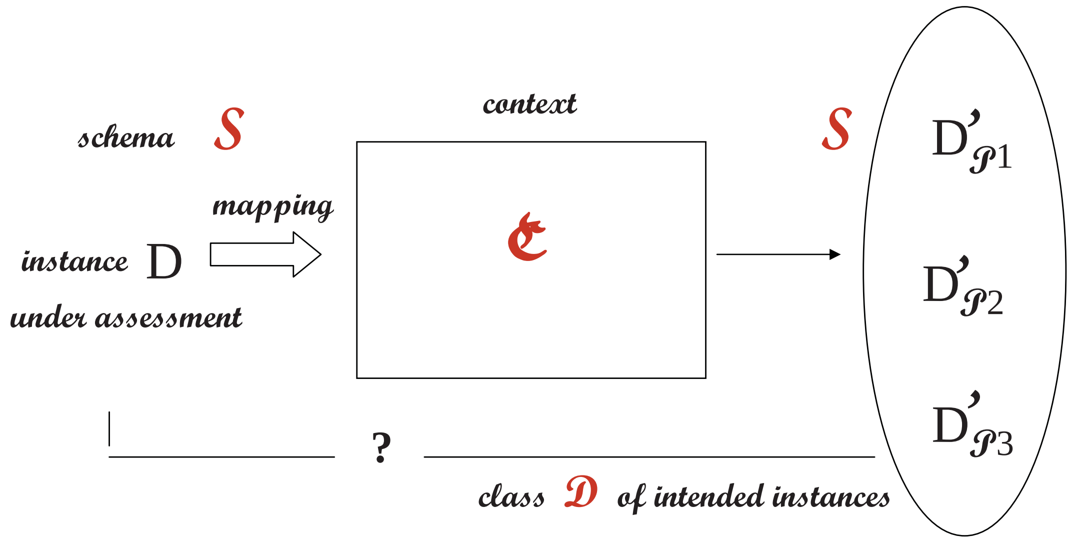
(B) D in logical connection with contextual data



In principle several versions of D can be obtained at the contextual level

Depending on the mapping, assumptions about the sources of data (completeness?), availability of (partial) data at the context, etc.

Quality criteria are imposed at the contextual level as before



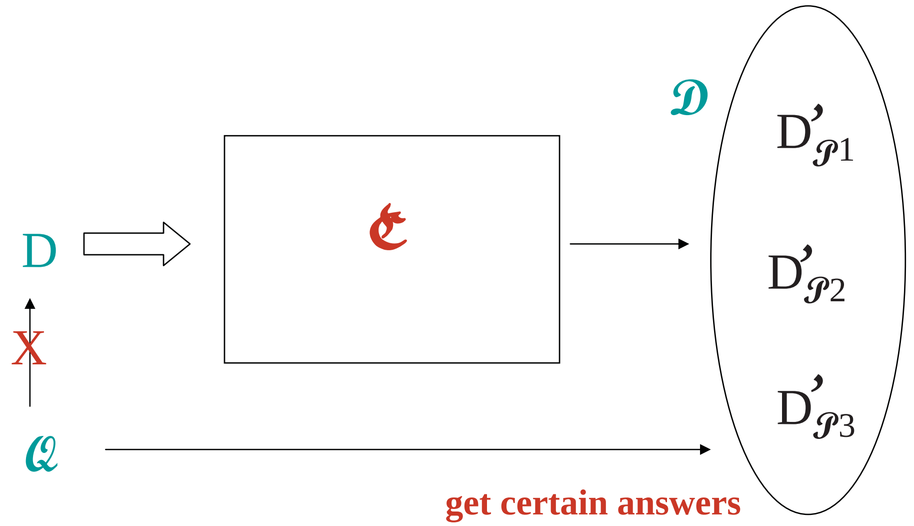
Quality of D can be measured through a distance to a class $\mathcal{D}$ of quality versions of it

Collection of instances on the RHS reflect "uncertainty" in D due to dirtiness

This framework opens the ground for "quality query answering"

Given a query $\mathcal{Q}$ posed to original, dirty D

Quality answers from D to $\mathcal{Q}$ are certain wrt class $\mathcal{D}$



Issues:

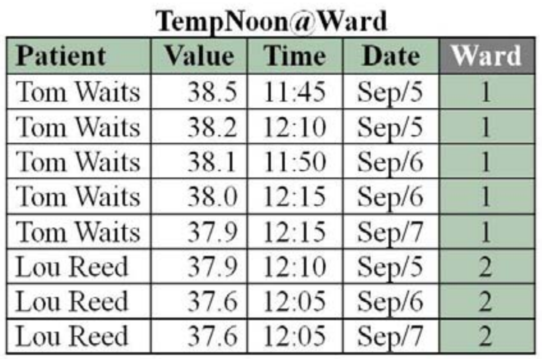
- Data cleaning vs. quality query answering
- Reminiscent of CQA
- Computation of quality answers

Multidimensional Contexts: The Idea

Temperature data at a hospital

Doctor requires temperatures taken with oral thermometer

Doctor expects this to be reflected in the table, but the latter does not contain the information to make this assessment



TempNoon@Ward

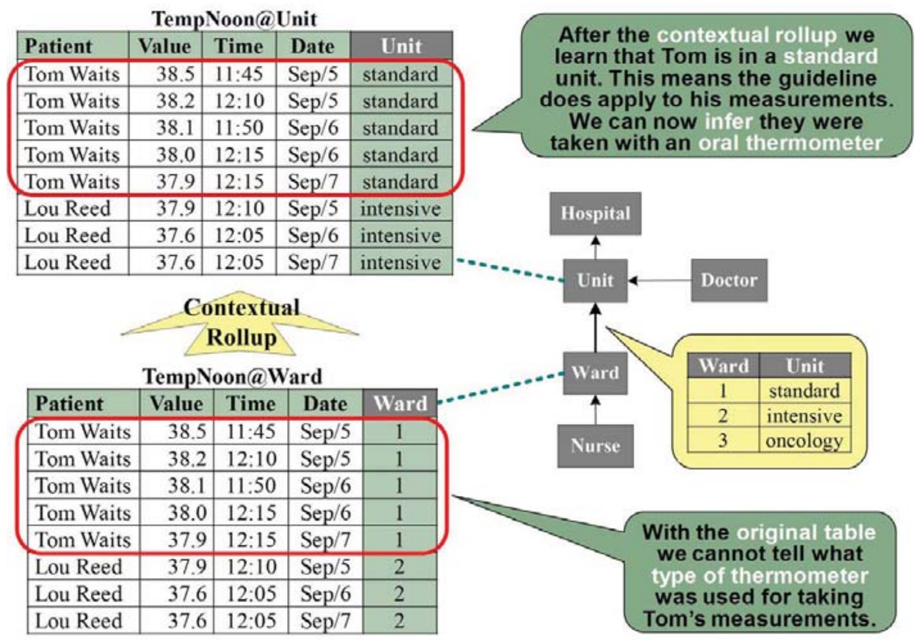
Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

An external context can provide that information, making it possible to assess the given data

The database under assessment is mapped into the context, for further data quality analysis, imposition of quality requirements, and cleaning

We can see the context as an ontology, containing:

(A) A MD data model/instanc


After the contextual rollup we learn that Tom is in a standard unit. This means the guideline does apply to his measurements. We can now infer they were taken with an oral thermometer
Hospital
Unit
Doctor
Ward
Nurse
Contextual Rollup
With the original table we cannot tell what type of thermometer was used for taking Tom's measurements.

TempNoon@Unit

Patient	Value	Time	Date	Unit
Tom Waits	38.5	11:45	Sep/5	standard
Tom Waits	38.2	12:10	Sep/5	standard
Tom Waits	38.1	11:50	Sep/6	standard
Tom Waits	38.0	12:15	Sep/6	standard
Tom Waits	37.9	12:15	Sep/7	standard
Lou Reed	37.9	12:10	Sep/5	intensive
Lou Reed	37.6	12:05	Sep/6	intensive
Lou Reed	37.6	12:05	Sep/7	intensive

TempNoon@Ward

Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

Ward	Unit
1	standard
2	intensive
3	oncology

(B) Information such as a hospital guideline:

"Temperatures of patients in standard care units are taken with oral thermometers"

In the form of a rule (hard or default) or a hard constraint

Can be taken advantage of after/through upward navigation

• Contextual information is commonly of a multi-dimensional nature

In the example, about the hierarchical hospital structure

Other dimensions could be easily considered, e.g. time

• Contextual, dimensional navigation for rolling-up/drilling-down used to access and generate missing information at certain levels

• We embed Hurtado-Mendelzon (HM) MD data models/instances in ontological contexts

A multidimensional context is generated for dimensional and finer-granularity data quality assessment

• Dimensions had not been considered in contextual for data quality analysis in [Bertossi et al. BIRTE'10]

Dimensions are naturally associated to contexts

Contexts with Dimensions: Work in Progress

(Join work with Mostafa Milani and Sina Ariyan)

Basis: HM model [Hurtado & Mendelzon, 2005]

We extend the HM model [Maleki et al., AMW'12]

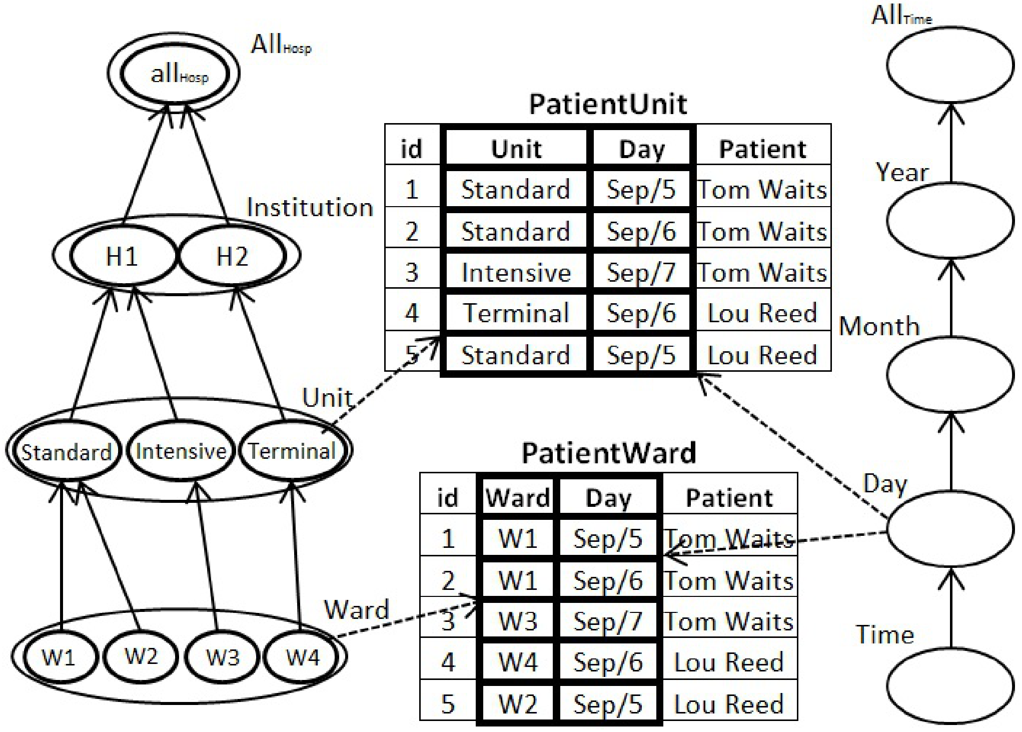
Informally, some of the new ingredients in MD contexts:

- Dimensions as in the HM
- Categorical relations: Generalize fact tables

  Not necessarily numerical values, linked to different levels of dimensions, possibly incomplete
- Dimensional rules: Generate data where missing
- Dimensional constraints

  Constraints on (combinations of) categorical relations, involve values from dimension categories

Example:



PatientUnit

id	Unit	Day	Patient
1	Standard	Sep/5	Tom Waits
2	Standard	Sep/6	Tom Waits
3	Intensive	Sep/7	Tom Waits
4	Terminal	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	W4	Sep/6	Lou Reed
5	W2	Sep/5	Lou Reed

Ward and *Unit*: categories of *Hospital* dimension

$UnitWard(unit,ward)$: a parent/child relation

PatientWard: categorical relation with *ward* and *day* categorical attributes taking values from dimension categories

Categorical relations are subject to dimensional constraints:

- A referential constraint restricting units in *PatientUnit* to elements in the *Unit* category, as a negative constraint

$$\bot \ \leftarrow \ \mathit{PatientUnit}(u, d; p), \neg \mathit{Unit}(u)$$

- *"All thermometers used in a unit are of the same type"*:

$$t = t' \ \leftarrow \ \mathit{Thermometer}(w, t; n), \mathit{Thermometer}(w', t'; n'), \mathit{UnitWard}(u, w), \mathit{UnitWard}(u, w') \qquad \text{An EGD}$$

$\mathit{Thermometer}(\mathit{ward}, \mathit{thermometertype}; \mathit{nurse})$ is categorical relation, t, t' variables for categorical attributes

- *"No patient in intensive care unit on August /2005"*:

$$\bot \ \leftarrow \ \mathit{PatientWard}(w, d; p), \mathit{UnitWard}(\texttt{Intensive}, w), \mathit{MonthDay}(\texttt{August}/2005, d)$$

Dimensional rules:

• Data in *PatientWard* generate data about patients for higher-level categorical relation *PatientUnit*

$$\begin{aligned} PatientUnit(u, d; p) \ \leftarrow \ & PatientWard(w, d; p), \\ & UnitWard(u, w) \end{aligned}$$

Since relation schemas "match", no existential quantifier in the head needed

Rule is used to navigate from *PatientWard.Ward* upwards to *PatientUnit.Unit* via *UnitWard*

• Once at the level of *Unit*, it is possible to take advantage of a guideline -in the form of a rule- stating that:

"Temperatures of patients in a standard care unit are taken with oral thermometers"

Information about thermometers and patients at *Unit* level that that can be used there and both at the *Ward* level

• Data in categorical relation *WorkingSchedules* generate data in categorical relation *Shifts*

WorkingSchedules

Unit	Day	Nurse	Type
Intensive	Sep/5	Cathy	cert.
Standard	Sep/5	Helen	cert.
Standard	Sep/6	Helen	cert.
Standard	Sep/9	Mark	non-c.

$\searrow$

Shifts

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

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

Captures a 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"*

In second rule, head has existential variable z for missing values for *shift* attribute

Rule can be used for downward navigation

A unit may drill-down to more than one ward, e.g. Standard unit is connected to wards W1 and W2, generating more than one tuple in Shifts

A query to *Shifts* asks for dates when Mark was working in ward W2

The query has no answer with the extensional data in *Shifts*

The last tuple in *WorkingSchedules* implies that Mark has shifts in both W1 and W2 on Sep/9

This date would be an answer obtained via downward navigation from the Standard unit to its wards (W1 and W2)

The example shows downward navigation is necessary for query answering, by propagating data in *WorkingSchedules* (at the unit level) down to *Shifts* (at the lower ward level)

Datalog$\pm$ as Representation Language

We use Datalog$\pm$ as our representation language [Cali et al., 2009]

An extension of Datalog for ontology building with efficient access to underlying data sources

An alternative (with additional positive features) to DL-Lite for ontology-based access to databases

Our approach to representation of MD contexts is general and systematic

It has the following general form:

- Negative constraints capturing referential constraints from categorical attributes to categories:

$$\bot \ \leftarrow \ R_i(\bar{e}_i;\bar{a}_i), \neg K(e)$$

$e, \bar{e}_i$ stand for categorical attributes, R_i a categorical predicate, and K a category predicate

- Dimensional constraints as EGDs or negative constraints:

$$\begin{aligned} x = x' \ \leftarrow \ & R_i(\boldsymbol{\bar{e}_i};\bar{a}_i), ..., R_j(\boldsymbol{\bar{e}_j};\bar{a}_j),\\ & D_n(e_n, e_n'), ..., D_m(e_m, e_m') \end{aligned}$$

$$\begin{aligned} \bot \ \leftarrow \ & R_i(\boldsymbol{\bar{e}_i};\bar{a}_i), ..., R_j(\boldsymbol{\bar{e}_j};\bar{a}_j),\\ & D_n(e_n, e_n'), ..., D_m(e_m, e_m') \end{aligned}$$

D_i are parent-child predicates

- Dimensional rules as TGDs:

$$\exists \bar{a}_z\ R_k(\bar{e}_k; \bar{a}_k) \quad \leftarrow \quad R_i(\bar{\boldsymbol{e}}_{\boldsymbol{i}}; \bar{a}_i), ..., R_j(\bar{\boldsymbol{e}}_{\boldsymbol{j}}; \bar{a}_j),$$
$$D_n(e_n, e'_n), ..., D_m(e_m, e'_m).$$

Existential quantifiers (possibly not needed) over non-categorical attributes, which may get labeled nulls as values

Join variables in bodies of TGDs only for categorical attributes

Upward or downward navigation captured by joins between categorical predicates and parent-child predicates in bodies

Properties of MD Ontologies

- Datalog$\pm$ is a family of languages with different syntactic restrictions on rules and their interaction to guarantee tractability

- Our Datalog$\pm$ MD ontologies become weakly-sticky Datalog$\pm$ programs [Cali et al., AIJ'12]

It is crucial that join variables in TGDs are for categorical attributes (a finite number of values can be taken by them, the category members)

- The separability condition on the (good) interaction between TGDs and EGDs becomes application dependent [Cali et al., ER'11]

However, if EGDs have categorical head variables, separability holds (as on page 21)

Separability implies decidability of conjunctive query answering

• The chase (that forwards propagates data through rules) may not terminate

Weak Stickiness guarantees tractability of conjunctive query answering (QA): only a "small", initial portion of the chase has to be inspected

Boolean conjunctive QA is tractable for weakly-sticky Datalog$\pm$ ontologies

The same applies to open conjunctive QA

• As opposed to sticky Datalog$\pm$, for weakly-sticky Datalog$\pm$ there is no general first-order query rewriting methodology

> That is, rewriting of conjunctive queries into FO queries in terms of underlying DB predicates

Query Answering on MD Ontology

We have developed a deterministic algorithm DeterministicWS-QAns for boolean conjunctive QA from MD ontologies

It is based on a non-deterministic P-time algorithm WeaklySticky-QAns for weakly-sticky Datalog$\pm$ [Cali et al., AIJ'12]

WeaklyStickyQAns:

- Builds an accepting resolution proof schema, a tree-like structure
- It shows how query atoms are entailed from extensional data
- Applies top-down backtracking for finding accepting resolution proof schemas

- Runs in polynomial time in the size of the extensional database

- For MD ontologies that support only upward navigation (which can be syntactically checked), we have a FO rewriting methodology for conjunctive QA

  In this case, the ontology may still not be sticky

Our algorithms are rather proofs of concept

Development of scalable polynomial time algorithms for open conjunctive queries with massive data is ongoing work

Downward Navigation and Categorical Attributes

TGDs as on pages 23 and 27 can be used for "deterministic" downward navigation: only values for non-categorical attributes are created, with determinism wrt. the categories involved

In some applications there may be incomplete data about the categorical attributes

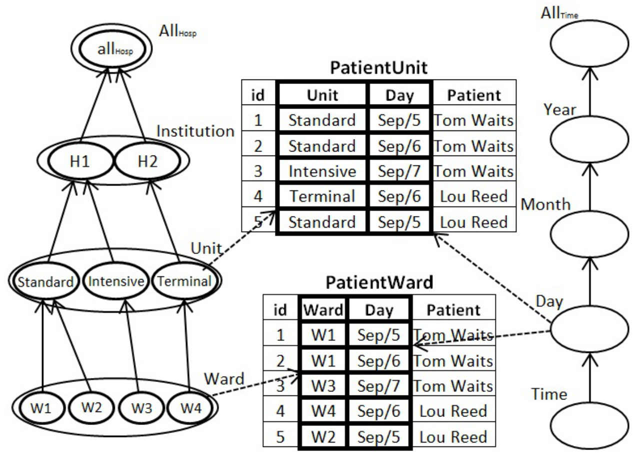
Existential quantifications over categorical variables may be needed

Example: Categorical relation *DischargePatients*, linked to *Institution*, with data about patients leaving the hospital

DischargePatients

Inst.	Day	Patient
H1	Sep/9	Tom Waits
H1	Sep/6	Lou Reed
H2	Oct/5	Elvis Costello

$\longrightarrow$



PatientUnit

id	Unit	Day	Patient
1	Standard	Sep/5	Tom Waits
2	Standard	Sep/6	Tom Waits
3	Intensive	Sep/7	Tom Waits
4	Terminal	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	W4	Sep/6	Lou Reed
5	W2	Sep/5	Lou Reed

Query on *PatientUnit* about the dates that 'Elvis Costello' was in a unit at institution 'H2'

No answer directly from *PatientUnit* (as derived from *Patient-Ward*)

If each patient is in a (only one) unit, *DischargePatient* can generate data downwards for *PatientUnit*

Knowledge about the unit (a category value) at the lower level is uncertain:

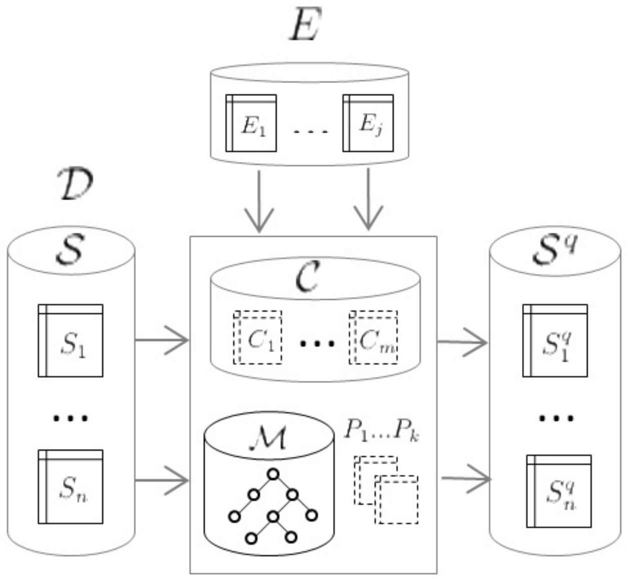
$$\exists u\ \mathit{InstitutionUnit}(i, u), \mathit{PatientUnit}(u, d; p) \ \leftarrow \ \mathit{DischargePatients}(i, d; p)$$

With rules of this kind, an MD ontology is still weakly-sticky: no infinite loops, only a limited number of new nulls can be generated with the chase

EGDs with only categorical attributes in heads do not guarantee separability anymore, and becomes application dependent

MD Contexts and Quality Query Answering: The Gist

The Datalog± MD ontology $\mathcal{M}$ becomes part of the context for data quality assessment



The original instance $\mathcal{D}$ is to be assessed or cleaned through the context

By mapping $\mathcal{D}$ into the contextual schema/instance $\mathcal{C}$

In the context:

- Contextual predicates C_i
- Predicates P_i specifying single quality requirements
- $\mathcal{S}^q$ copy of schema $\mathcal{S}$: S_i^q clean version of original S_i, specified using $\mathcal{C}, \mathcal{P}$ and $\mathcal{M}$

Example: We want quality answers to the query about Tom's temperatures:

$$\mathcal{Q}(t,p,v) \leftarrow \mathit{Measurements}(t,p,v), p = \texttt{Tom Waits}, \\ \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.$$

Quality requirements are not captured by this query; we expect:

"Body temperatures of *Tom Waits* for *September 5* around noon taken by a *certified nurse* with a thermometer of *brand B1*"

Table *Measurements* does not contain information about nurses or thermometers

Contextual data must be taken into account, such as categorical relation *PatientUnit* and the guideline

"Temperature measurement for patients in a standard care unit are taken with thermometers of brand B1"

According to the general contextual approach DQA, table (or better predicate) $Measurement$ has to be logically connected to the context

As a "footprint" of a "broader" contextual table that is given or built in the context, in this case, one with information about thermometer brands (b) and nurses' certification status (y):

$$\begin{aligned} Measurement'(t,p,v,y,b) \quad \leftarrow \quad & Measurement^c(t,p,v), \\ & TakenByNurse(t,p,n,y), \\ & TakenWithTherm(t,p,b) \end{aligned}$$

$Measurement^c$ is contextual version of $Measurement$ (e.g. the latter mapped into the context)

If we want quality measurements data, we impose the required conditions:

$$Measurement^q(t,p,v) \leftarrow Measurement'(t,p,v,y,b), \; y = \texttt{Certified}, \; b = \texttt{B1}$$

The auxiliary predicates above:

$$\begin{aligned} TakenByNurse(t,p,n,y) &\leftarrow WorkingSchedules(u,d;n,y), \\ &\quad DayTime(d,t), PatientUnit(u,d;p) \\ TakenWithTherm(t,p,b) &\leftarrow PatientUnit(u,d;p), \\ &\quad DayTime(d,t), b = \texttt{B1}, u = \texttt{Standard} \end{aligned}$$

($DayTime$ is parent/child relation in *Time* dimension)

The second definition is capturing the guideline above

To obtain quality answers to the original query, we pose to the ontology the new query:

$$Q^q(t,p,v) \ \leftarrow Measurements(t,p,v)^q, \ p = \texttt{Tom Waits}, \\ \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.$$

Answering it triggers dimensional navigation, when requesting data for categorical relations $PatientUnit$ and $WorkingSched$-$ules$

Conclusions

We have described in general terms how to:

- Build contexts for data quality enforcement
- Extend them with dimensions via a MD ontology
- Specify MD ontologies in Datalog±
- Use query answering from those ontologies for quality query answering
- MD Contexts are interesting and useful also outside data quality concerns
- They are logical extensions of multidimensional databases

We have identified some properties of MD ontologies:

- They fall into well-known and tractable syntactic classes of Datalog$\pm$
- Query answering can be done in polynomial time in data
- For some classes of EGDs in the ontology, separability wrt. TGDs holds
- For ontologies with some classes of TGDs, related to navigation direction, FO query rewriting of conjunctive queries is guaranteed
- They extend previous approaches to "dimensional query answering", e.g. to querying database with taxonomies
  [Martinenghi & Torlone; ER'10]

Main ongoing task: Improvement of query answering algorithm