Semantic Constraints for Data Quality Assessment and Cleaning

Leopoldo Bertossi
Carleton University
Ottawa, Canada
(Faculty Fellow IBM CAS)
A database may not satisfy a given set of integrity constraints

What is the consistent data in an inconsistent database?

What are the consistent answers to a query posed to an inconsistent database?

A mathematically precise definition was needed

In (Arenas,Bertossi,Chomicki; PODS99) such a characterization was provided

Intuitively, the consistent data in an inconsistent database $D$ is invariant under all minimal ways of restoring $D$'s consistency

That is, consistent data persists across all the minimally repaired versions of the original instance: the repairs of $D$
**Example:** For the instance $D$ that violates 

**FD:** $\text{Name} \rightarrow \text{Salary}$

<table>
<thead>
<tr>
<th>Employee</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>5K</td>
<td></td>
</tr>
<tr>
<td>page</td>
<td>8K</td>
<td></td>
</tr>
<tr>
<td>smith</td>
<td>3K</td>
<td></td>
</tr>
<tr>
<td>stowe</td>
<td>7K</td>
<td></td>
</tr>
</tbody>
</table>

Two possible (minimal) **repairs** if only deletions/insertions of whole tuples are allowed: $D_1$, resp. $D_2$

<table>
<thead>
<tr>
<th>Employee</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>5K</td>
<td></td>
</tr>
<tr>
<td>smith</td>
<td>3K</td>
<td></td>
</tr>
<tr>
<td>stowe</td>
<td>7K</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>8K</td>
<td></td>
</tr>
<tr>
<td>smith</td>
<td>3K</td>
<td></td>
</tr>
<tr>
<td>stowe</td>
<td>7K</td>
<td></td>
</tr>
</tbody>
</table>

$(\text{stowe, 7K})$ persists in all repairs: it is consistent information  

$(\text{page, 8K})$ does not; actually it participates in the violation of **FD**
A consistent answer to a query $Q$ from a database $D$ is an answer that can be obtained as a usual answer to $Q$ from every possible repair of $D$ wrt $IC$ (a given set of ICs)

- $Q_1 : Employee(x, y)?$
  Consistent answers: $(smith, 3K), (stowe, 7K)$

- $Q_2 : \exists y Employee(x, y)?$
  Consistent answers: $(page), (smith), (stowe)$

CQA may be different from classical data cleaning!

However, CQA is relevant for data quality; an increasing need in business intelligence

It also provides concepts and techniques for data cleaning
Next DBMSs should provide more flexible, powerful, and user-friendly mechanisms for dealing with semantic constraints. In particular, they should allow to be posed queries requesting for consistent data; and answer them.

Why not **an enhanced SQL**?

```sql
SELECT Name, Salary
FROM Employee
CONS/W FD: Name -> Salary;
```

(FD not maintained by the DBMS)

**Paradigm shift:** ICs are constraints on query answers, not on database states!
Depending on the ICs and the queries, tractable and intractable cases for CQA have been identified.

For some tractable cases, query rewriting algorithms have been developed:

\[
Q(x, y): \text{Employee}(x, y) \quad \rightarrow \quad Q'(x, y): \text{Employee}(x, y) \land \neg \exists z(\text{Employee}(x, z) \land z \neq y)
\]

For higher-complexity cases, specifications of repairs by means of logic programs with stable model semantics can be used.

CQA becomes querying (as usual) a logic program, say a Datalog program with possible complex extensions.
There are some implemented systems for CQA

- FO query rewriting (when possible)
- Graph-theoretic algorithmic methods
  Repairs can be implicitly represented as, e.g. maximal independent sets in a conflict graph or hypergraph
- Based on optimized (disjunctive) logic programs with stable model semantics (plus DLV)

More recently: Increasing interest in computing a single, “good” repair, or even an approximate repair

As a form of data cleaning wrt IC violation or semantic problems
A natural application: **Virtual data integration**

No way to enforce consistency on the sources

Inconsistencies have to be solved on-the-fly, at query time
Many problems in CQA addressed in the last few years

- Query rewriting mechanisms
- Compact representations of all DB repairs: Graph-theoretic, logic programs with stable model semantics, disjunctive databases, models of theories in non-classical logics, etc.
- Identification of tractable vs. non-tractable cases
- Applications in virtual data integration, PDMS, etc.
- Implementations
New Kinds of Constraints: Data Quality

Integrity constraints (ICs) have been around for a long time. They are used to capture the application semantics in the data model and database. They have been studied in general and have wide application in data management. A large body of research has been developed, in particular fundamental research. Methodologies for dealing with ICs are quite general and have broad applicability. Database repairing and CQA are newer contributions in this direction.
On the other side:

**Data quality assessment (DQ) and data cleaning (DC) have been mostly: Ad-hoc, rigid, vertical, and application-dependent activities**

There is a lack of fundamental research in data quality assessment and data cleaning

Things are starting to change ...

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

Proposed and studied by the Edinburgh DB group around Wenfei Fan
Conditional Dependencies (CDs)

Example: Database relation with FDs:

$FD_1: [CC, AC, Phone] \rightarrow [Street, City, Zip]$ 

$FD_2: [CC, AC] \rightarrow [City]$ 

<table>
<thead>
<tr>
<th>CC</th>
<th>AC</th>
<th>Phone</th>
<th>Name</th>
<th>Street</th>
<th>City</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>131</td>
<td>1234567</td>
<td>mike</td>
<td>mayfield</td>
<td>NYC</td>
<td>EH4 8 LE</td>
</tr>
<tr>
<td>44</td>
<td>131</td>
<td>3456789</td>
<td>rick</td>
<td>crichton</td>
<td>NYC</td>
<td>EH4 8LE</td>
</tr>
<tr>
<td>01</td>
<td>908</td>
<td>3456789</td>
<td>joe</td>
<td>mtn ave</td>
<td>NYC</td>
<td>07974</td>
</tr>
</tbody>
</table>

FDs are satisfied, but they are “global” ICs

They may not capture natural data quality requirements, as related to specific data values (important in data quality assessment and data cleaning)

What about a conditional functional dependency (CFD)?
\[
CFD_1: \quad [CC = 44, Zip] \rightarrow [Street]
\]
Conditional in that the FD of \textit{Street} upon \textit{Zip} applies when the country code is 44

Not satisfied anymore, and data cleaning may be necessary ...

More generally, CDs are like classical ICs with a \textit{tableau} for forced data value associations

\[
CFD_2:
[CC = 44, AC = 131, Phone] \rightarrow [Street, City='EDI', Zip]
\]
When \( CC = 44, AC = 131 \) hold, the FD of \textit{Street} and \textit{Zip} upon \textit{Phone} applies, and the city is \textit{‘EDI’}

Not satisfied either ...

CQA and database repairs have been investigated for CFDs
[Kolahi, Lakshmanan], [Beskales, Ilyas, Golab], ...
Conditional Inclusion Dependencies:

\[ \text{Order}(\text{Title}, \text{Price}, \text{Type} = \text{'book'}) \subseteq \text{Book}(\text{Title}, \text{Price}) \]

It can be expressed in classical FO predicate logic:

\[ \forall x \forall y \forall z (\text{Order}(x, y, z) \land z = \text{'book'} \rightarrow \text{Book}(x, y)) \]

Still a classic flavor ...

And semantics ...
**Matching Dependencies (MDs)**

MDs are related to **Entity Resolution** (ER)

ER is a classical, common and difficult problem in data cleaning

It is about discovering and matching records that represent the same entity in the application domain

Again, several ad hoc mechanisms have been proposed

ER, and DC in general, are fundamental for data analysis and decision making in BI

Particularly crucial in data integration, and even more in virtual data integration (VDI)

In VDI, DC and ER have to be made on-the-fly, at query time
MDs express and generalize ER concerns

They specify attribute values that have to be made equal under certain conditions of similarity for other attribute values

Example: Schema $R_1(X, Y), R_2(X, Y)$

$$\forall X_1X_2Y_1Y_2(R_1[X_1] \approx R_2[X_2] \rightarrow R_1[Y_1] = R_2[Y_2])$$

When the values for attributes $X_1$ in $R_1$ and $X_2$ in $R_2$ in two tuples are similar, then the values in those two tuples for attribute $Y_1$ in $R_1$ and $Y_2$ in $R_2$ must be made equal (matched)

($R_1$ and $R_2$ can be same predicate)

$\approx$: Domain-dependent similarity relation

Introduced by W. Fan et al. (PODS 2008, VLDB 2009)
Although declarative, MDs have a procedural feel and a dynamic semantics.

An MD is satisfied by a pair of databases \((D, D')\):

- \(D\) satisfies the antecedent, and \(D'\), the consequent, where the matching is realized.

But this is local, one-step satisfaction ...
Our research: [ICDT'11, KR’12, ..., LID’11, SUM’12, DATALOG 2.0’12]

• Alternative, refined semantics for MDs

• Investigation of the dynamic semantics

• Definition and computation of clean instances
  (there may be several of them)

• Definition of “clean query answering”, and computational methods to obtain them

• Comparisons between clean instances wrt MDs and database repairs wrt FDs

• Query rewriting methodologies for clean query answering
  (in Datalog plus aggregation)

• ASP-based specification of clean instances
MDs as originally introduced do not say how to identify values

\[ \forall X_1 X_2 Y_1 Y_2 (R_1[X_1] \approx R_2[X_2] \rightarrow R_1[Y_1] \dot{=} R_2[Y_2]) \]

We have considered the two directions:

- With matching functions (MFs) (ICDT 2011, etc.), and
- Without MFs (LID 2011, etc.)
Matching Dependencies with MFs

“similar name and phone number \( \Rightarrow \) identical address”

<table>
<thead>
<tr>
<th>( D_0 )</th>
<th>name</th>
<th>phone</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>(613)123 4567</td>
<td>Main St., Ottawa</td>
<td></td>
</tr>
<tr>
<td>J. Doe</td>
<td>123 4567</td>
<td>25 Main St.</td>
<td></td>
</tr>
</tbody>
</table>

\[ ⇓ \]

<table>
<thead>
<tr>
<th>( D_1 )</th>
<th>name</th>
<th>phone</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>(613)123 4567</td>
<td>25 Main St., Ottawa</td>
<td></td>
</tr>
<tr>
<td>J. Doe</td>
<td>123 4567</td>
<td>25 Main St., Ottawa</td>
<td></td>
</tr>
</tbody>
</table>

A dynamic semantics!

\[
m_{\text{address}}(\text{Main St.}, \text{Ottawa}, \text{25 Main St.}) := \text{25 Main St.}, \text{Ottawa}
\]

Addresses treated as strings or objects, i.e. sets of pairs attribute/value

(Join work with Solmaz Kolahi and Laks Lakshmanan)
Semantics of MDs: [W. Fan et al., VLDB’09]

\[ \varphi : \quad R_1[X_1] \approx R_2[X_2] \rightarrow R_1[Y_1] \vdash R_2[Y_2] \]

\((D, D') \models \varphi\) if for every \(R_1\)-tuple \(t_1\) and \(R_2\)-tuple \(t_2\):

\[ t_1[X_1] \approx t_2[X_2] \text{ in } D \implies t_1[Y_1] = t_2[Y_2] \text{ in } D' \]

\(D'\) is stable if \((D', D') \models \Sigma\) (a set of MDs)

Dirty instance \(D \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots \Rightarrow D'\)

\[ \uparrow \]

stable, clean instance!

- How are the MDs enforced?
- Can we expect that \((D, D') \models \Sigma\)? (too strong)
Matching Functions: Some ingredients

- Set of MDs $\Sigma$
- For every attribute $A$ with $\text{Dom}_A$
  - A similarity relation $\approx_A \subseteq \text{Dom}_A \times \text{Dom}_A$ reflexive and symmetric
  - A matching function $m_A : \text{Dom}_A \times \text{Dom}_A \rightarrow \text{Dom}_A$
    idempotent, commutative, and associative

Induces a semilattice with partial order defined as

$$a \preceq_A a' \iff m_A(a, a') = a'$$

Least upper bound operator coincides with matching function

$$\text{lub}\{a, a'\} = m_A(a, a')$$
$a \preceq_A a'$ can be thought of in terms of information contents

A semantic-domination lattice is created (… “domain theory”)

- Domain-level lattice

- Tuple-level partial order:

$$t_1 \preceq t_2 \iff t_1[A] \preceq_A t_2[A] \text{ (f.a. } A)$$

- Relation-level partial order

$$D_1 \sqsubseteq D_2 \iff \forall t_1 \in D_1 \exists t_2 \in D_2 t_1 \preceq t_2$$
Instances can be “reduced” by eliminating tuples that are dominated by others

**Theorem:** The set of reduced instances with \( \sqsubseteq \) forms a lattice

Relevant for comparison of sets of query answers seen as instances ...
Clean Instances:
\[ \varphi : R_1[X_1] \approx R_2[X_2] \rightarrow R_1[A_1] \cong R_2[A_2] \]

One step of chase: Enforcing \( \varphi \) on \( D \Rightarrow D' \)

- In \( D \), \( t_1[X_1] \approx t_2[X_2] \), but \( t_1[A_1] = a_1 \neq t_2[A_2] = a_2 \)
- In \( D' \), replace them with \( m_A(a_1, a_2) \)

Clean instance: Stable instance resulting from chase

\[ D_0 \Rightarrow D_1 \Rightarrow \ldots \Rightarrow D_{\text{clean}} \]

Theorem: Matching functions idem, comm, assoc give us:

(a) Chase termination after polynomial number of steps

(b) \( D_0 \subseteq D_1 \subseteq \ldots \subseteq D_{\text{clean}} \)
In general:

- There could be multiple clean instances
- It may not hold \((D_0, D_{\text{clean}}) \models \Sigma\)

For two special cases:

- Similarity-preserving matching functions
  \[ a \approx a' \implies a \approx m_A(a', a'') \]

- Interaction-free MDs
  - There is a unique clean instance \(D_{\text{clean}}\), and
  - \((D_0, D_{\text{clean}}) \models \Sigma\)
Example: Assume only these two resolved instances for $D$:

<table>
<thead>
<tr>
<th>$D_1^c$</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
</tr>
<tr>
<td>J. Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
</tr>
<tr>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_2^c$</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>Main St., Ottawa</td>
<td></td>
</tr>
<tr>
<td>J. Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
</tr>
<tr>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
</tr>
</tbody>
</table>

(A) We can compute/choose one of them, possibly using some additional requirement

(B) We consider a certain semantics: What is true is what is invariant across (in common in) all resolved instances

Query $Q$: SELECT * FROM R

- $Certain(Q, D) = \{\langle Jane Doe, 25 Main St., Vancouver \rangle \}$
  
  Does not take underlying domain into account, very strict

- $Certain(Q, D) = \{\langle Jane Doe, 25 Main St., Vancouver \rangle, 
  \langle John Doe, Main St., Ottawa \rangle, \langle J. Doe, 25 Main St. \rangle \}$

  Takes domain (MFs) into account
Clean answers to a query $Q$:

- Certain answers: $\text{glb}_{\subseteq} \{ Q(D) \mid D \text{ clean instance} \}$
- (Possible answers: $\text{lub}_{\subseteq} \{ Q(D) \mid D \text{ clean instance} \}$)

With two clean instances as above:

<table>
<thead>
<tr>
<th>$D_1^c$</th>
<th>name</th>
<th>address</th>
<th>$D_2^c$</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>25 Main St., Ottawa</td>
<td>John Doe</td>
<td>Main St., Ottawa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J. Doe</td>
<td>25 Main St., Ottawa</td>
<td>J. Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query $Q'$: $\pi_{\text{address}} (\sigma_{\text{name}="J. Doe"}(R))$

Certain $= \text{glb}_{\subseteq} \{ Q'(D_1^c), Q'(D_2^c) \} = \{25 \text{ Main St.}\}$

(Possible $= \{25 \text{ Main St., Ottawa}, 25 \text{ Main St., Vancouver}\}$)

Theorem: Computing certain clean answers is coNP-complete
Monotonicity?

\( D \subseteq D' \) is not set-inclusion

A query \( Q \) is monotone if: \( D \subseteq D' \implies Q(D) \subseteq Q(D') \)

Why not taking advantage of lattice-theoretic domain structure when posing queries?

Proposition: A positive relational algebra query composed of \( \pi, \times, \cup, \sigma_{a \leq A}, \sigma_{A_1 \sqsubseteq A_2} \) is monotone, where

\[
\begin{align*}
t \in \sigma_{a \leq A}(D) & :\iff a \preceq t[A] \\
t \in \sigma_{A_1 \sqsubseteq A_2}(D) & :\iff \text{glb}\{t[A_1], t[A_2]\} \neq \bot
\end{align*}
\]

We obtain monotone queries

Monotonicity and clean query answering?
<table>
<thead>
<tr>
<th>$D^c_1$</th>
<th>name</th>
<th>address</th>
<th>$D^c_2$</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>John Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
<td>John Doe</td>
<td>Main St., Ottawa</td>
</tr>
<tr>
<td></td>
<td>J. Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
<td>J. Doe</td>
<td>25 Main St., Vancouver</td>
</tr>
<tr>
<td></td>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
</tr>
</tbody>
</table>

Query $Q'' : \pi_{\text{name}}(\sigma_{\text{"25 Main St." } \preceq \text{address}}(R))$ (monotone)

$Q''(D^c_1) = \{\text{John Doe, J. Doe, Jane Doe}\}$

$Q''(D^c_2) = \{\text{J. Doe, Jane Doe}\}$

$\text{Certain}(Q'') = \text{glb}_{\subseteq} \{Q''(D^c_1), Q''(D^c_2)\} = \{\text{J. Doe, Jane Doe}\}$

$Q''(\text{glb}_{\subseteq} \{D^c_1, D^c_2\}) = Q''(\{\langle \text{John Doe , Main St., Ottawa}\rangle, \langle \text{J. Doe , 25 Main St.}\rangle, \langle \text{Jane Doe , 25 Main St., Vancouver}\rangle\})$

$= \text{Certain}(Q'')$
In general, for the class $\mathcal{D}$ of clean instances

**Proposition:** For a monotone query:

$\exists D \downarrow$

$\exists Q(\text{glb}_{\subseteq}\{D \mid D \in \mathcal{D}\}) \subseteq \text{glb}_{\subseteq}\{Q(D) \mid D \in \mathcal{D}\}$

$\exists \text{lub}_{\subseteq}\{Q(D) \mid D \in \mathcal{D}\} \subseteq Q(\text{lub}_{\subseteq}\{D \mid D \in \mathcal{D}\})$

- Under-approximate certain answers by $Q(D\downarrow)$
- Over-approximate possible answers by $Q(D\uparrow)$

Adding heuristics to chase to obtain approximations to $D\downarrow, D\uparrow$?
Recent/Ongoing Research and Look Ahead

• Logic programs (ASPs) for clean QA in presence of MDs

LPs specify clean instances

LP-based declarative formulations of known ER algorithms, e.g. Swoosh

• Make query posing/answering sensitive to semantic-domination lattice

• Approximate query answering based on relaxation using semantic domination lattice

• Computing clean answers from data subject to MDs (without physically cleaning it)

Query rewriting, approximations, ...

[KR 2012, Datalog 2.0, 2012]
• ER and **virtual data integration**

Declarative specifications of ER could be compiled into query answering!

Virtual data integration is a natural application scenario

**On-the-fly ER!**
• ER and data exchange under schema mappings:

Traditionally: Materialize a (good) target instance $\mathcal{J}$ with:

$$(\mathcal{I}, \mathcal{J}) \models \Sigma_{st} \text{ and } \mathcal{J} \models \Sigma_t$$

Now: also apply MDs when shipping data from $\mathcal{I}$ to $\mathcal{J}$

ER at data exchange time ...
A table containing data about the temperatures of patients at a hospital

<table>
<thead>
<tr>
<th>Patient</th>
<th>Value</th>
<th>Time</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom Waits</td>
<td>38.5</td>
<td>11:45</td>
<td>Sep/5</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>38.2</td>
<td>12:10</td>
<td>Sep/5</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>38.1</td>
<td>11:50</td>
<td>Sep/6</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>38.0</td>
<td>12:15</td>
<td>Sep/6</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>37.9</td>
<td>12:15</td>
<td>Sep/7</td>
</tr>
</tbody>
</table>

Is this quality data?
If not, is there anything to clean? What?

(Join work with Flavio Rizzolo)
We do not know ... It depends ...

Actually the table is supposed to contain *temperature measurements for Tom taken at noon by a certified nurse with an oral thermometer*

Is this quality data? We still do not know ...

Maybe we can say something about the time

Maybe good enough for the time to be “around noon” (meaning?)

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
- assess the data
- on that basis, support data cleaning
- etc. (see below)
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
Actually, the connection between $\mathcal{T}$ and $\mathcal{C}$ is established through: connection predicates and 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 seen as a logical theory, e.g. Reiter’s logical reconstruction of a relational DB
More concretely:

(A)

Data in \( \mathcal{C} \) (including \( D \)) is analyzed/cleaned

According to additional data available in or accessible from \( \mathcal{C} \); and quality criteria defined in \( \mathcal{C} \)
A new version of $D$ is obtained and can be compared with $D$ for quality assessment
D is mapped into a contextual ontology

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 $D$ of quality versions of it

This framework opens the ground for “quality query answering”
Given a query $Q$ posed to original, dirty $D$

Quality answers from $D$ to $Q$ are certain wrt class $D$

Issues:
- Data quality assessment vs. quality query answering
- Data cleaning vs. quality query answering
- Computation of quality answers
Multidimensional Contexts

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

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
Hospital guideline:

“The temperature of patients in standard care units have to be taken with an oral thermometer”

Captured by means of a rule (hard, or possibly, default rule)

Or a hard constraint

The information in the context is commonly of a multidimensional nature

We embed (an extension of) the Hurtado-Mendelzon model for MDDBs into our ontological context
A specification of the hierarchical/dimensional hospital structure

Other dimensions could be easily considered, generating multidimensional (MD) contextual information, for additional and finer-granularity data quality assessment.
Contextual roll-up can be used to access missing information at certain level, by lattice navigation.

Mechanisms for querying database with taxonomies could be applied/embedded (Martinenghi & Torlone; ER10)

Many interesting issues open ...
Extra Slides

On MDs:

- Under-approximate certain answers by $Q(D_{\downarrow})$
- Over-approximate possible answers by $Q(D_{\uparrow})$

Adding heuristics to chase to obtain $D_{\downarrow}, D_{\uparrow}$?

Under cleaning: not enforcing interacting MDs

Over cleaning: assuming matching functions are similarity preserving

Computing or approximating those two instances using $D$ alone?
Associativity of MF is a natural assumption, not only because without it we can’t have a lattice and termination of chase, etc., but also because it makes sense in any entity resolution process such as ours.

That is, when during the process we identify three or more data values that are representing the same entity, the result of collapsing them into one value should not depend on the order in which we visit those values.

If the aggregate function to be used is not associative, e.g. the average, we can always use union, and apply the aggregate function at the very end (average for instance).