Carleton
UNIVERSITY

Some Recent Fundamental Approaches to Data Quality Assessment and Cleaning

Leopoldo Bertossi⋆
Carleton University
Ottawa, Canada

⋆: Faculty Fellow IBM CAS. Also affiliated to University of Concepción, Chile

1. Characterizing Consistent Data wrt ICs

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: $Name \rightarrow Salary$

$Employee$	$Name$	$Salary$
	$Page$	5K
	$Page$	8K
	$Smith$	3K
	$Stowe$	7K

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

$Employee$	$Name$	$Salary$
	$Page$	5K
	$Smith$	3K
	$Stowe$	7K

$Employee$	$Name$	$Salary$
	$Page$	8K
	$Smith$	3K
	$Stowe$	7K

$(Stowe, 7K)$ persists in all repairs: it is consistent information

$(Page, 8K)$ does not; actually it participates in the violation of FD

A consistent answer to a query $\mathcal{Q}$ from a database D is an answer that can be obtained as a usual answer to $\mathcal{Q}$ from every possible repair of D wrt IC (a given set of ICs)

- $\mathcal{Q}_1 : Employee(x, y)?$

  Consistent answers: $(Smith, 3K), (Stowe, 7K)$

- $\mathcal{Q}_2 : \exists y Employee(x, y)?$

  Consistent answers: $(Page), (Smith), (Stowe)$

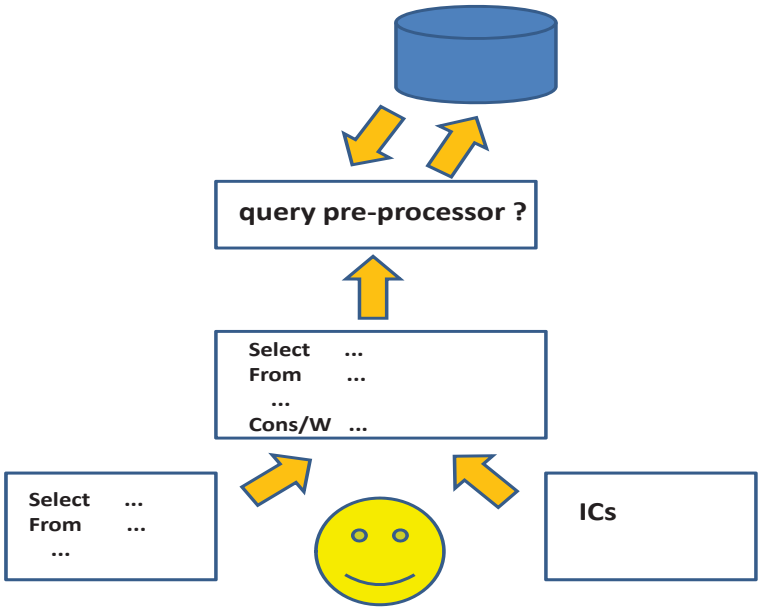
CQA may be different from classical data cleaning!

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 friendlier mechanisms for dealing with semantic constraints

In particular, they should allow to be posed queries requesting for consistent data; and answer them

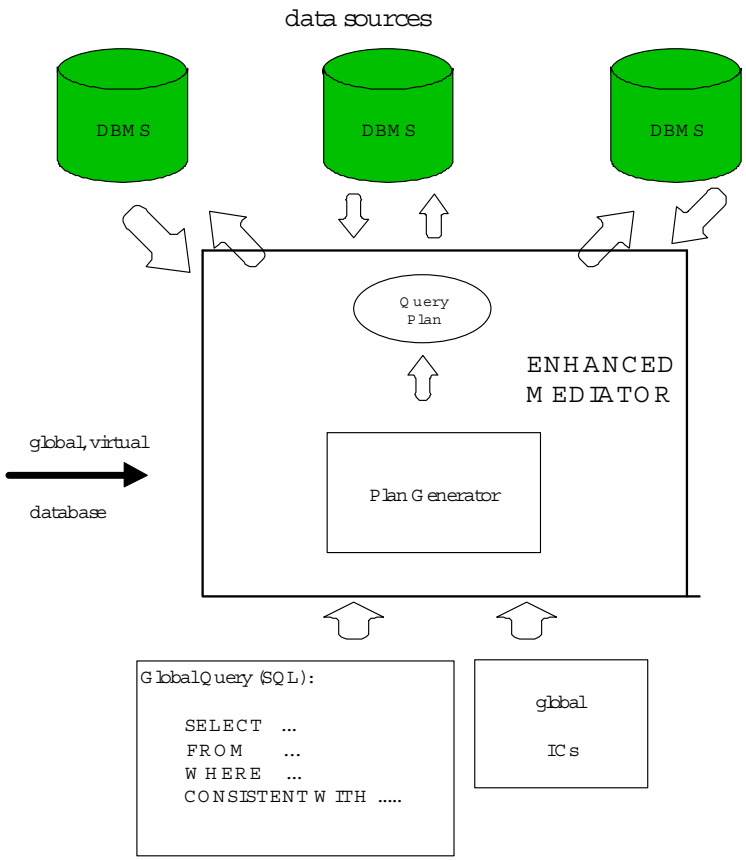


Why not an enhanced SQL?

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!



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 around CQA have been 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, peer data management systems, etc.
- Implementations

2. New Kinds of Constraints

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

CQA is a contribution in this direction

On the other side, data quality (DQ) assessment and data cleaning (DC) have been: ad hoc activities, rigid, vertical, and application dependent

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

E.g. conditional dependencies and matching dependencies

They provide generic languages for expressing quality concerns

Suitable for specifying adaptive and generic DQ/C mechanisms

Entity Resolution:

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

Declarative specifications for ER could be in principle compiled into query answering!

Matching Dependencies:[1]

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 $R_1(X, Y), R_2(X, Y)$

$$\forall X_1 X_2 Y_1 Y_2 (R_1[X_1] \approx R_2[X_2] \implies R_1[Y_1] \equiv R_2[Y_2])$$

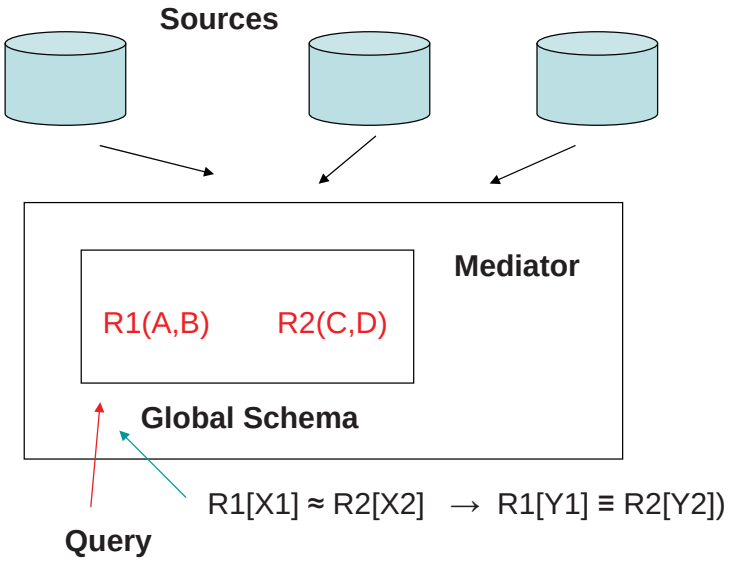
(read: have to be made equal)

Introduced by W. Fan et al. (PODS 2008, VLDB 2009)

We have developed and investigated their dynamic semantics

Also computing clean answers from data subject to MDs (without physically cleaning it): Query rewriting, approximations, ...

[1] Join work with Solmaz Kolahi and Laks Lakshmanan



Again, virtual data integration is a natural application scenario

On-the-fly ER!

MDs as originally introduced do not say how to identify values

We have considered the two directions: With matching functions (ICDT 2011) and without them (LID 2011, with Jaffer Gardezi)

Matching Dependencies with Matching Functions

- Record matching: identifying data referring to the same entity.
- Matching dependencies (MDs): declarative rules for record matching (Fan et al. in 2009)

"similar name and phone number $\Rightarrow$ identical address"

D_0	*name*	*phone*	*address*
	John Doe	(613)123 4567	Main St., Ottawa
	J. Doe	123 4567	25 Main St.

$\Downarrow$

D_1	*name*	*phone*	*address*
	John Doe	(613)123 4567	25 Main St., Ottawa
	J. Doe	123 4567	25 Main St., Ottawa

dynamic semantics!

Semantics of MDs:

$$\varphi : R_1[X_1] \approx R_2[X_2] \;\rightarrow\; R_1[Y_1] \equiv R_2[Y_2]$$

A pair of instances $(D, D') \models \varphi$ if for every R_1-tuple t_1 and R_2-tuple t_2

$$t_1[X_1] \approx t_2[X_2] \text{ in } D \;\Rightarrow\; \begin{array}{l} t_1[Y_1] = t_2[Y_2] \text{ in } D' \\ t_1[X_1] \approx t_2[X_2] \text{ in } D' \end{array}$$

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

Dirty instance $D \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots\ldots \Rightarrow$ stable D'

- How are the MDs enforced?
- Can we expect that $(D, D') \models \Sigma$? (condition too strong)

Matching Functions: Some ingredients

- Set of MDs Σ
- For every attribute A with Dom_A
  - A similarity relation $\approx_A \subseteq Dom_A \times Dom_A$
  - A matching function $\boldsymbol{m}_A : Dom_A \times Dom_A \rightarrow Dom_A$ idempotent, commutative, and associative.

Induces a semilattice with partial order defined as

$$a \preceq_A a' \;\Leftrightarrow\; \boldsymbol{m}_A(a, a') = a'$$

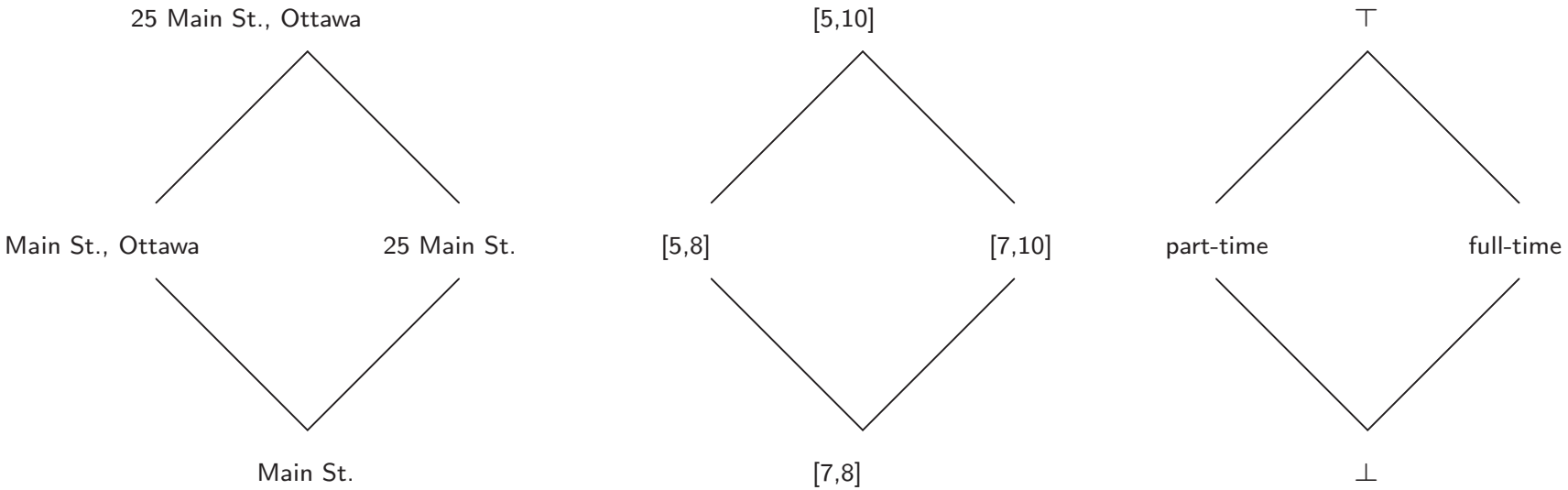
Least upper bound operator coincides with matching function

$$lub\{a, a'\} = \boldsymbol{m}_A(a, a')$$

Can be thought of in terms of information contents

Semantic Domination Lattice:

- Domain-level lattice



- Tuple-level partial order $t_1 \preceq t_2 \Leftrightarrow t_1[A] \preceq_A t_2[A]$
- Relation-level partial order

$$D_1 \sqsubseteq D_2 \Leftrightarrow \forall t_1 \in D_1 \; \exists t_2 \in D_2 \; t_1 \preceq t_2$$

Theorem: The set of *reduced* instances with $\sqsubseteq$ ordering forms a lattice (get rid of dominated elements)

Clean Instances:

$$\varphi : R_1[X_1] \approx R_2[X_2] \rightarrow R_1[A_1] \equiv R_2[A_2]$$

One step of chase: enforcing φ once 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

$$D_0 \Rightarrow D_1 \Rightarrow \ldots \Rightarrow D_{clean}$$

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

(a) Chase termination after polynomial number of steps

(b) $D_0 \sqsubseteq D_1 \sqsubseteq \ldots \sqsubseteq D_{clean}$

Query Answering:

Clean answer to a query $\mathcal{Q}$ by providing two bounds

- certain answer: $glb_{\sqsubseteq}\{\mathcal{Q}(D) \mid D \text{ clean instance}\}$
- possible answer: $lub_{\sqsubseteq}\{\mathcal{Q}(D) \mid D \text{ clean instance}\}$

D	*name*	*address*
	John Doe	25 Main St., Ottawa
	J. Doe	25 Main St., Ottawa
	Jane Doe	25 Main St., Vancouver

D'	*name*	*address*
	John Doe	Main St., Ottawa
	J. Doe	25 Main St., Vancouver
	Jane Doe	25 Main St., Vancouver

Query $\mathcal{Q}$: $\pi_{address}(\sigma_{name=\text{“J. Doe”}}(R))$

certain $= \{\text{25 Main St.}\}$

possible $= \{\text{25 Main St., Ottawa}, \text{25 Main St., Vancouver}\}$

Theorem: Computing clean answer is coNP-complete

Ongoing Research:

- Approximate query answering based on relaxation using semantic domination lattice
- Logic programs for clean QA in presence of matching dependencies

3. Contexts and Data Quality

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

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

It may be good enough that the time is "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
- etc. (see below)

Contexts So Far

We find the term "context" in several places in computer science: databases, semantic web, KR, mobile applications, ...

Usually used for "*context aware* ... search, databases, applications, devices, ..."

Most of the time there is no explicit notion of context, but only some algorithms that take into account (or into computation) some contextual notions

Usually, time and geographic location, i.e. particular *dimensions*, and not much beyond

In our opinion, there is a lack of research in the area

A precise and formalized notion of context is still missing

There has been some research:

- Contexts in ontologies and semantic web

  Contexts are left implicit and logic programs are used to bridge them

- Contexts in KR

  They are denoted at the object level and a theory specifies their properties and dynamics

  It is possible to talk about things holding in certain (named) contexts

- Contexts in data management

  Usually in connection with specific dimensions of data, like time and place

Contexts: A Vision

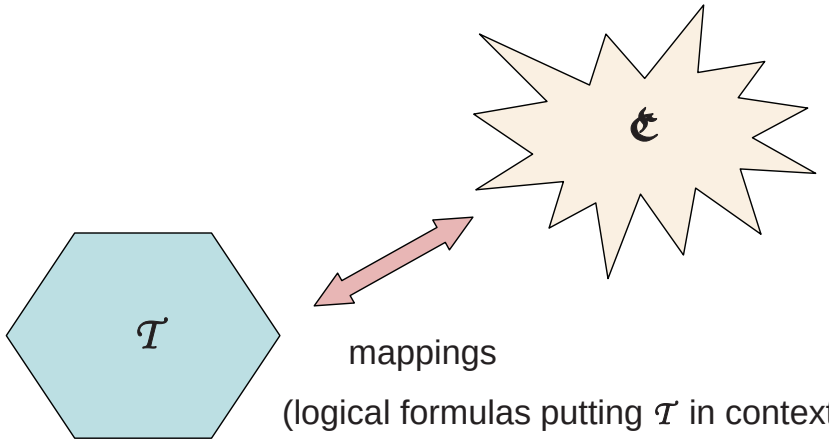
A general theory of context has still to be developed

In particular for applications in data management,

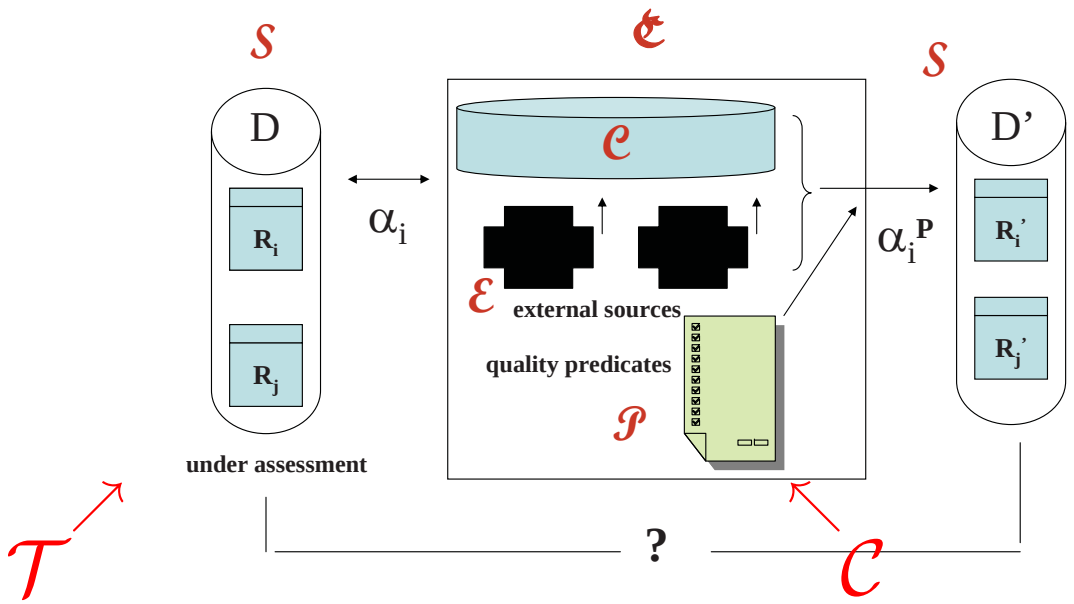
We envision it as follows:

- 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, $\mathcal{C}$
  $\mathcal{T}$ and $\mathcal{C}$ share some predicate symbols
- Actually, the connection between $\mathcal{T}$ and $\mathcal{C}$ is established through: connection predicates and mappings

In general:



In our data quality scenario: (VLDB BIRTE WS 2010)

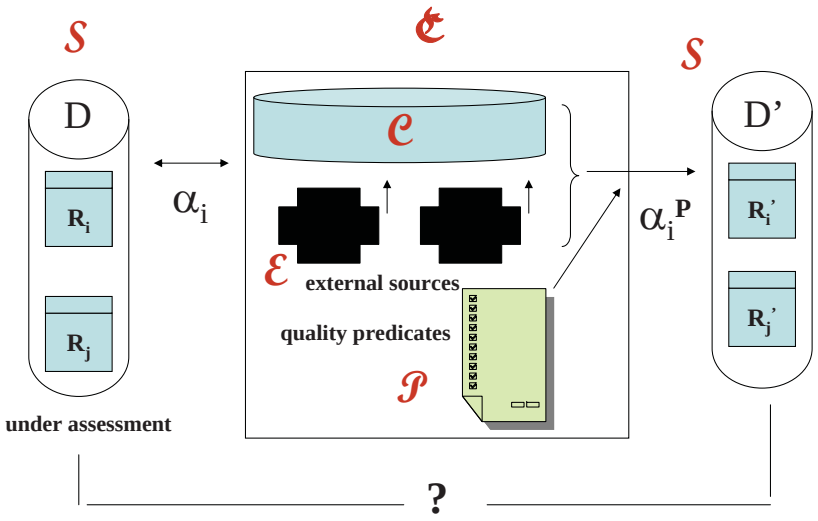


In the general formalization, the mappings and the way they are processed (reasoned about and from) have to be such that they enable what we expect from a context, i.e support for the following tasks

- Capturing and narrowing down semantics
  - By defining in $\mathcal{C}$ predicates that are used in $\mathcal{T}$ (e.g. "time close to noon")
  - Contributing in $\mathcal{C}$ with additional constraints for predicates used in $\mathcal{T}$, e.g. integrity constraints for table TempNoon)
- Term disambiguation (related to meaning)
- Dimensions for analysis and understanding of $\mathcal{T}$'s knowledge (generalizing multidimensional DBs, DWHS)

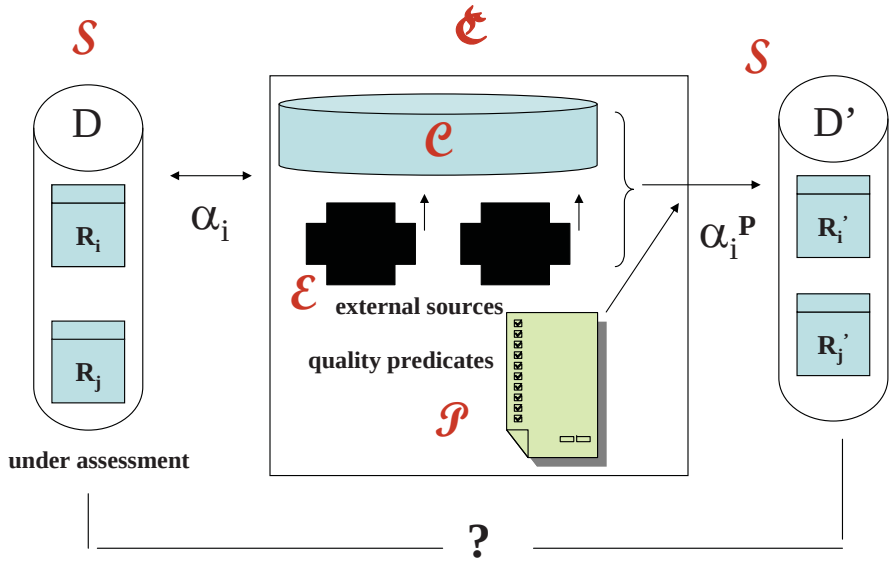
- Specifying and using notions of relevance
- Explanation, diagnosis, causality
- Capturing commonsense assumptions and practices
- Assessment, e.g. quality

Contexts in Data Quality Assessment



- Instance $\mathcal{S}$ is under assessment
- Context $\mathcal{C}$ (including $\mathcal{E}$, $\mathcal{P}$) on the RHS, as a virtual/(semi)materialized data integration system

- The α_i in between are the mappings, like in VDISs or data exchange
- The C_i are contextual predicates/relations
- There are mappings to external sources E_i and quality predicates/relations P_i



- D' (with schema as in $\mathcal{S}$) contains "ideal" contents for relations in D, as views
- Predicates in D' can be materialized through data in the R_i and additional massage via $\mathcal{C}$ (mapping composition at work)

- Quality-aware (QA) query answering about (or from) $\mathcal{S}$ can be done on top of D'

  Techniques for query answering in VDISs can be applied (specially if D' is not materialized)

- Quality assessment of D can be done by comparing its contents with D'

  A particular case of QA query answering

  There are some measures to distance between database instances (for the same schema, as here)

Multidimensional Contexts (ongoing research)

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

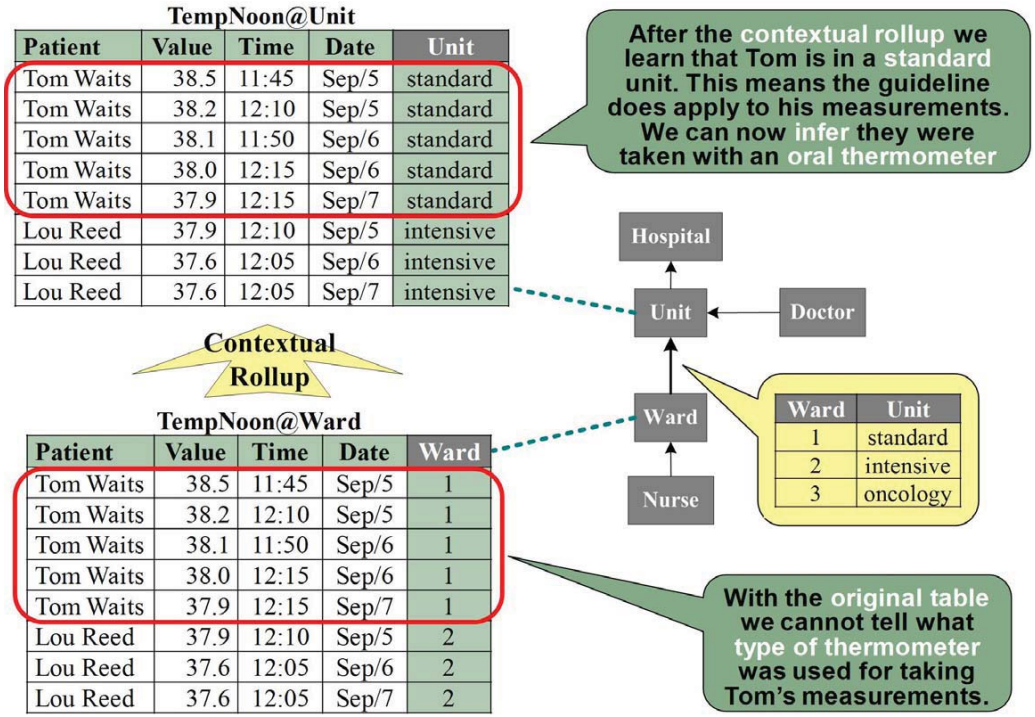
Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

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 analysis and cleaning

The information in the context is commonly of a multidimensional nature:

- Hospital guideline: *the temperature of patients in standard care units have to be taken with an oral thermometer*



- A specification of the hierarchical and dimensional hospital structure

Other dimensions could be easily considered, generating multidimensional (MD) contextual information, for additional and finer-granularity data quality assessment

Conclusions

The general formalization and computational use of contexts is still an open problem

Many aspects of contexts have to be taken into account and modeled

Ours is a long term general research

Also in terms of applications to data quality assessment and cleaning

We have sketched some first steps in this direction

Next steps have to do with: (a) Use of quality predicates, (b) Specification of *sense* (of data items) by imposing additional semantics, (c) Techniques for QA query answering

In (database centered, lower-level) data management, data quality assessment usually deals with problems arising from the acquisition and integration of data: typos, inaccuracy, incompleteness, inconsistency, etc.

At the other end, BI applications require data quality assessment at higher levels of abstraction, where subjectiveness, usefulness, sense, and interpretation play a central role

From a BI perspective, the meaning of the data, in a broad sense, and therefore its quality, are context dependent

In our broad and long term research we are investigating the role and use of contexts in data quality assessment and cleaning

With flexible, adaptive and generic data quality frameworks, solutions and tools in mind