Data Quality and Data Integration

Leopoldo Bertossi[⋆]

Carleton University
School of Computer Science
Ottawa, Canada

⋆: Faculty Fellow of the IBM Center for Advanced Studies

Data Integration

Capabilities to both generate and collect data and information have experienced an explosive growth

Advances in business data collection, e.g. from finance, retail and production devices, have generated a flood of data and information

For data to be useful, it has to be integrated

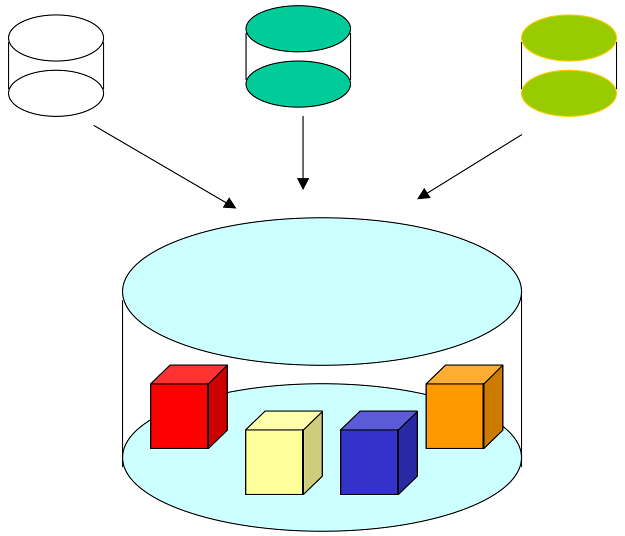
- Possibly from multiple and heterogeneous data sources
- Sources not necessarily structured as relational databases

  Increasing amount of semi- or non-structured data

Many interesting scientific and technical problems arise

For conceptual simplicity, DI approaches fall in one of two broad categories: materialized and virtual

Materialized Approaches



A basic assumption is that we have some sort of control on the data sources, to keep up-to-date integrated data

A new, physical database is created importing data from other data sources, usually from operational and transactional DBs

Data sources are independent and autonomous

Rather rigid in terms of participating sources

Data in the new database may be structured differently

E.g. relational tables at the sources, but “cubes” in the new DB

Cubes suggesting different dimensions of data, that give context to (usually) numerical data

E.g. units sold (a number) associated to: a product, a location, and a time (three dimensions)

Possible a collection of materialized views defined on the combination of the original data sources

But also historical information that may not be recoverable from the sources

Most common incarnation of the materialized approach: Data Warehouses (DWHs)

- Data Quality and Data Cleaning:

Under the materialized approach, data quality/cleaning has a “classic” flavor, as found in single, stand alone DBs

For BI applications, DWHs are expected to contain quality data

In particular, syntactically and semantically consolidated data ...

Consolidation and cleaning can be partially done already when data is moved from the sources to the DWH

ETL tools: Extract, Transport, Load ...

Still, after that, the DWH may still require data cleaning

- DWH and traditional DBs share the limitations of data cleaning solutions (cf. below)

- DWH can be done with relational technology

  The multidimensional view of data can be captured using suitable relational schemas

  However, dimensions are hierarchical

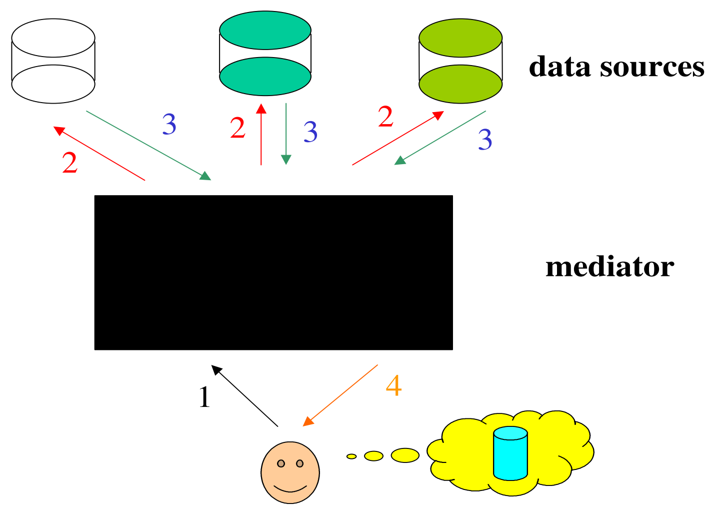
  Multiple relational integrity constraints (ICs), actually foreign-key constraints, become necessary

  To ensure proper navigational and summarizability properties

  These semantic constraints (among others) have to be kept satisfied

  Consistency maintenance becomes an issue (cf. below)

Virtual Data Integration



Data is virtually integrated from several autonomous and heterogeneous data sources

Increasingly relevant given the diversity, number, volatility, and dynamism of data sources!

Accessing them on-the-fly, as in their current status, may be crucial for decision making

The user interacts with a mediator, having the feeling of being interacting with a single, real database, with a global schema

Sources do not collaborate with each other

They only answer queries from the mediator

Sources cannot be updated from the mediator

The global schema could be relational, but sources may not be relational

But the latter could be wrapped as relational

The mediator has to create a query plan to answer the user query, which involves retrieving data from the sources

Data Quality and Data Cleaning

Data understanding and decision making heavily rely on high quality data

Much activity around data cleaning today, specially in the context of BI

Several conferences and research projects

Two important research areas in BI and Data Quality, in particular:

Assessment of Data Quality:

- Models of data quality?
- Specification of quality data?
- Data quality constraints?

  As an extension of usual (semantic) integrity constraints in databases (cf. below)
- Too many dimensions of data quality: accuracy, completeness, redundancy, freshness, consistency, etc.
- Quality predicates and languages for expressing quality concerns?
- Their evaluation against concrete instances?
- Characterization of the clean data in a (globally) dirty database?

Data Cleaning:

- Existing solutions are specific and vertical

  How to generalize from verticals?

  More general, flexible, adaptable, parameterizable solutions?

- Data cleaning is highly context dependent

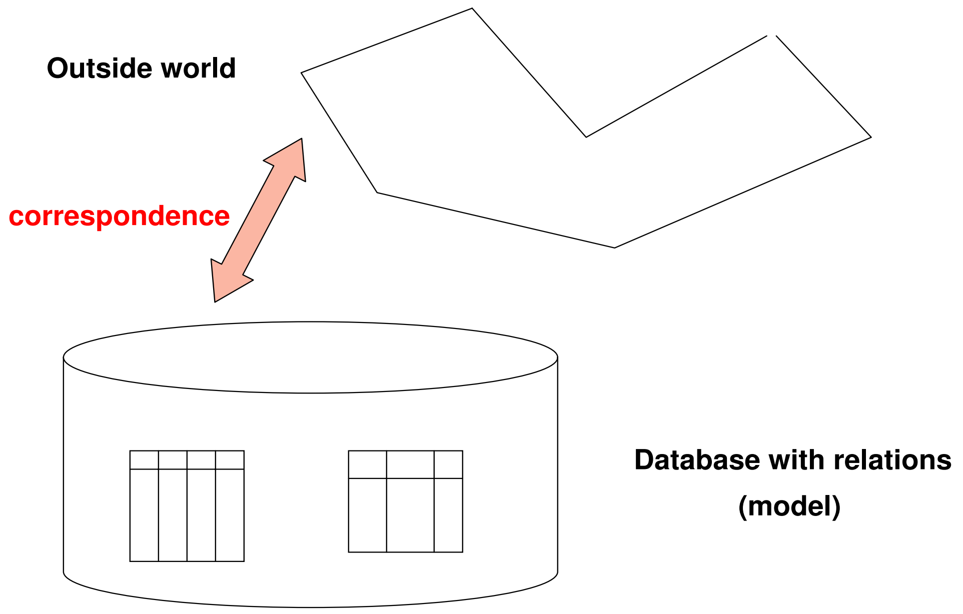
  Contexts could specify different domains, e.g. temporal, spatial, products, customers, etc.

  How to specify and automate use of contexts for data cleaning?

- How to derive, specify and run cleaning rules from all of the above and past experiences? (rule mining)

- How to obtain clean data on-the-fly, at query or application time (as opposed to cleaning the whole instance) (cf. below)

Consistency of Databases



A database instance D is a model of an outside reality

An integrity constraint on D is a condition to be satisfied by D in order to capture the semantics of the application domain

A set IC of integrity constraints (ICs) helps maintain the correspondence between D and that reality

ICs capture a certain aspect of data quality

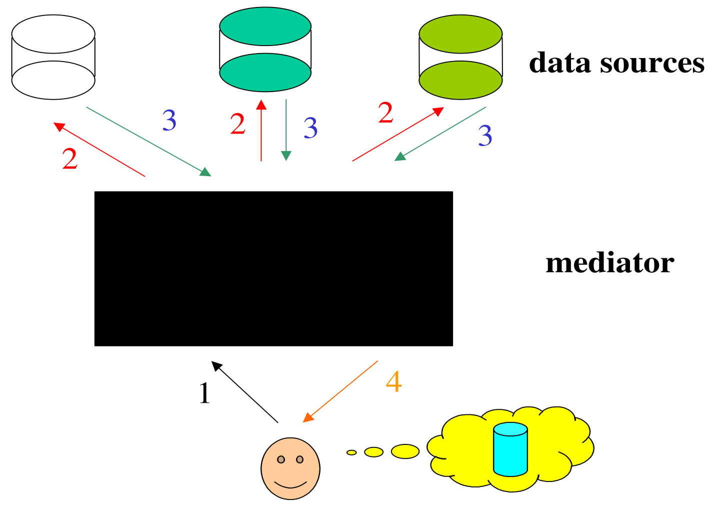
If D satisfies IC, we say that D is consistent

For several reasons a database may become inconsistent with respect to a given set of desirable ICs, e.g.

- Non-enforced ICs
- User constraints
- Legacy data without the currently required semantics
- Data integration

Local ICs $\xrightarrow{\surd}$

Global ICs $\xrightarrow{?}$



Sources may be consistent wrt their own, local ICs

What if we want to impose global ICs, i.e. on the global schema?

No way to maintain a set of global ICs satisfied!

Mediator has no control on the sources, and data is not integrated into one single repository

Traditional data cleaning or integrity restoration are not possible

Clean/Consistent data has to be obtained on-the-fly, at query answering time

Which is the consistent data in an inconsistent database (or integration system)?

Characterizing Consistent Data

If D is inconsistent, we might try to restore consistency

Restoring consistency wrt ICs is a form of data cleaning

Not always possible or desirable ...

Bringing D to a consistent state may be difficult, impossible, nondeterministic, undesirable, unmaintainable, etc.

We may have to live with inconsistent data ...

The database (the model) is departing from the outside reality that is being modeled

However, the information is not all semantically incorrect

Most likely most of the data in the database is still “consistent”

Idea: (a) Keep the database as it is

(b) Obtain semantically meaningful information at query time; dealing with inconsistencies on-the-fly

But before: Which is the consistent data in an inconsistent database?

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

The consistent data in an inconsistent database D can be defined as invariant under all minimal ways of restoring D's consistency

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:

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

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

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

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

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

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

  Consistent answers: $(Smith, \text{3K}), (Stowe, \text{7K})$

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

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

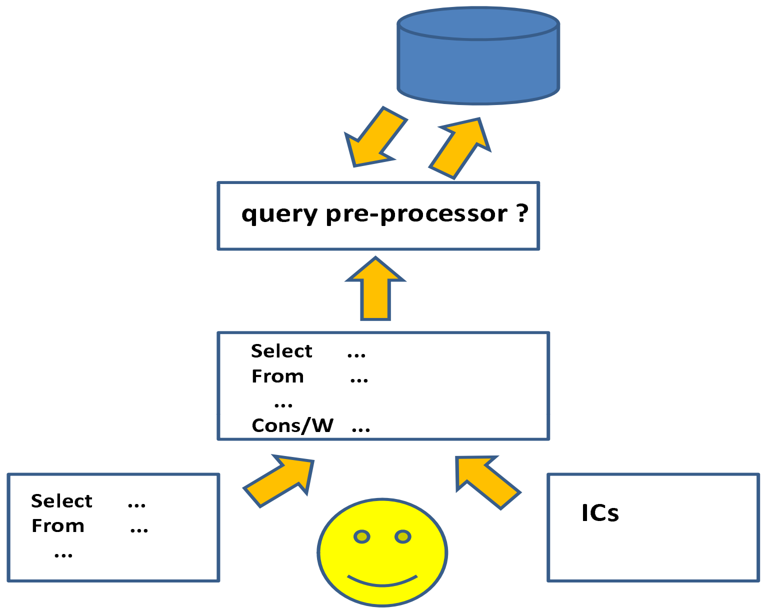
CQA may be different from classical data cleaning!

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

It also provides concepts, techniques, and approaches for/to 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!

What about computing CQA?

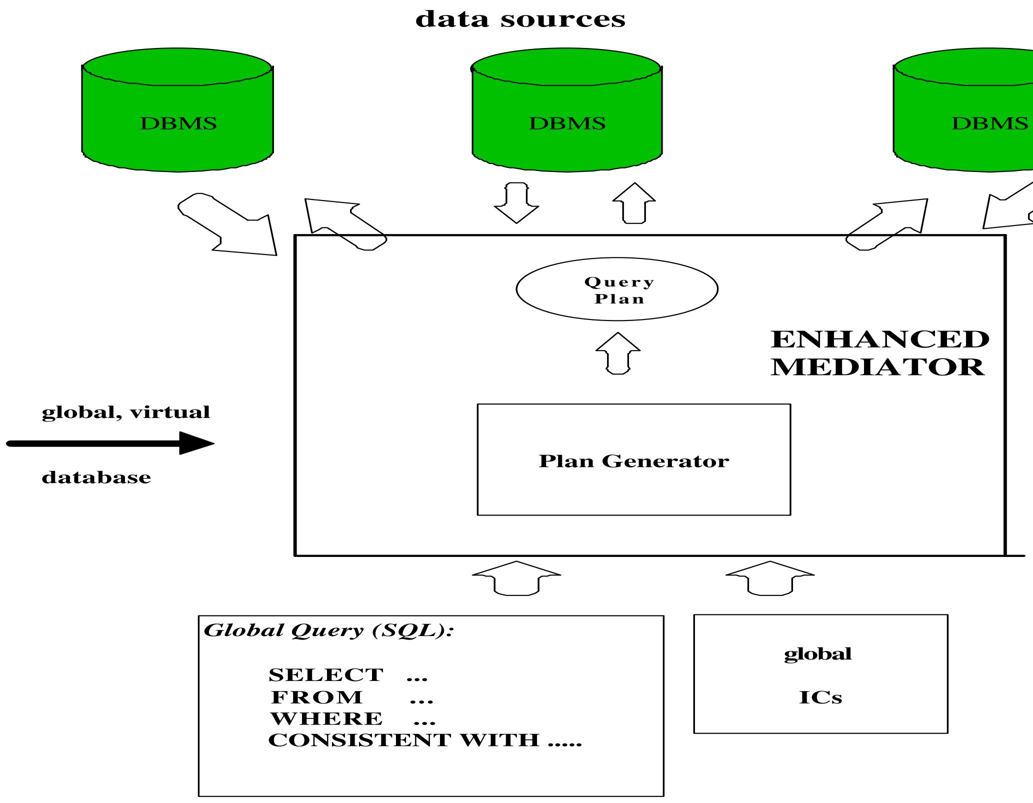
Query $\mathcal{Q}_1$ can be rewritten into a new query, actually an SQL query to be posed to the original database

```
SELECT      Name, Salary    ↦    SELECT      Name, Salary
FROM        Employee;            FROM        Employee
                                 WHERE       NOT EXISTS (
                                   SELECT *
                                   FROM    Employee E
                                   WHERE   E.Name = Name AND
                                           E.Salary <> Salary);
```

(retrieves employees with their salaries for which there is no other employee with the same name, but different salary)

Standard answers to this standard query from the original database are the consistent answers to query $\mathcal{Q}_1$

No explicit repair is needed to consistently answer this query!



Concepts and techniques for consistent query answering can be applied to virtual data integration

An enhanced query planner has to be able to use the global ICs

Data Quality Constraints

The ideas behind consistent query answering wrt ICs can also be applied to data quality and data cleaning

What about obtaining clean data (answers) at query or application time?

Consistent query answering is relative to a set of classical semantic ICs

The latter are easily specified in a declarative manner, e.g. using the relational calculus

What about specifying data quality concerns?

There is quite recent research addressing this important problem

Two dimensions of quality constraints (DQCs):

- What are they trying to express?
  What quality concerns are they expressing/capturing?
- What (logical) language can be used to express them?

Some Classes of DQCs:

1. Integrity Constraints:

These are the classic semantics constraints, like functional dependencies (FDs), inclusion dependencies, etc.

CC	AC	phn	name	street	city	zip
44	131	1234567	Mike	Mayfield	NYC	EH4 8 LE
44	131	3456789	Rick	Crichton	NYC	EH4 8LE
01	908	3456789	Joe	Mtn Ave	NYC	07974

$FD1 : [CC, AC, phn] \rightarrow [street, city, zip]$

$FD2 : [CC, AC] \rightarrow [city]$

Satisfied by the instance above, but may not capture natural data quality requirements ...

Need more?

2. Conditional FDs: (W. Fan et al.)

Motivation in data cleaning

Like classical FDs, but relative to certain data values

Referring to specific data values important in quality assessment/cleaning

$CFD1 : [CC = \text{`}44', zip] \longrightarrow [street]$

The FD should hold when the country code takes the value 44

3. Conditional Dependencies: (generalization of 2.)

A set of classical ICs comes with a tableau (or pattern) indicating forced data value associations

$CFD2 : [CC = \text{`}44', AC = \text{`}131', phn] \rightarrow [street, city = \text{`}EDI', zip]$

$CFD2 : [CC = `44', AC = `131', phn] \rightarrow [street, city = `EDI', zip]$

Like in 2., but under the indicated conditions, city must also take the value 'EDI'

We can also have conditional inclusion dependencies:

$$Order(title, price, type = `book') \subseteq Book(title, price)$$

These constraints still have a classic flavor and semantics, and can be expressed in relational calculus

Many classic problems related to dependencies re-emerge for conditional dependencies and are being actively investigated

4. Matching Dependencies:

A well-studied problem in data cleaning: detecting tuples that refer to the same real-world entity

Known as: duplicate detection, reference reconciliation, entity resolution, data fusion, etc.

Constraints can be used to express which values should be matched

Although declarative, they have a procedural feeling and a "dynamic" semantics

$$\bigwedge_j R_1[X_1[j]] \approx_j R_2[X_2[j]] \quad \rightarrow \quad R_1[Y] \rightleftharpoons R_2[Y]$$

Explicitly specify the matching actions leading to the "clean" instance

Interesting problems:

- Similarity predicates may (should) be context-dependent

  Evaluating them could need domain/context knowledge
- Matching operator could be domain/application/task dependent
- After the matching is specified,we can do the actual cleaning
- Or we could do it on-the-fly, without modifying the material instance

  Particularly appealing (necessary) in mediated data integration
- For answering query $\mathcal{Q}$, compile $\rightleftharpoons$ into $\mathcal{Q}$

  $\mathcal{Q} \mapsto \mathcal{Q}'$, with $\mathcal{Q}'$ a rewritten version of $\mathcal{Q}$

  Pose $\mathcal{Q}'$ to the original instance ...

5. Declarative Quality Concerns:

Explicitly expressing quality concerns in predicate logic, with quality attributes

They explicitly take into account the different dimensions of data quality

They introduce data quality predicates, e.g. Accuracy, Precision, Currency, Relevance, Completeness, Timeliness, Reliability, Believability

Those predicates are used as ingredients for a logical language that can be used to express quality requirements

Those predicates have to be specified/axiomatized depending upon the context or application domain

(L. Jiang, A. Borgida, J. Mylopoulos. ER 2008)

Conclusions

- Data quality and data cleaning are hot research subjects today

Much of this effervescent activity is motivated by their relevance to Business Intelligence

- Many challenges to data cleaning come from data integration

Virtual data integration is a particularly challenging application domain

Doing data cleaning on-the-fly is a possible way to address the problem

- Classic integrity constraints are still relevant for data quality/cleaning

However, they do not capture all the data quality concerns

- New forms of quality constraints are emerging and are being investigated

Classic ICs capture the semantics of consistent data

Quality constraints, like matching dependencies, capture the "semantics of dirty data"

THANK YOU!

`bertossi@scs.carleton.ca`

`www.scs.carleton.ca/~bertossi`