Entity Resolution with Matching Dependencies: Beyond Certainty

Leopoldo Bertossi*

Carleton University
School of Computer Science
Ottawa, Canada

$\star$: Faculty Fellow of the IBM CAS

Duplicate Resolution and MDs

A (relational) database may contain several representations of the same external entity

The database contains "duplicates", which is in general considered to be undesirable

The database has to be cleaned ...

The problem of duplicate- or entity-resolution is about:

(a) detecting duplicates, and

(b) merging duplicate representations into single ones

This is a classic and complex problem in data management, and data cleaning in particular

We concentrate mostly on the merging part of the problem

Relevance for BI

Data quality assessment and data cleaning are crucial subjects in business applications

Only with quality data we can do proper data analysis, learn from data, and correctly support decision making

We are flooded with data, and we do not always know how to:

- Make sense of data

  Understand, interpret, assign semantics, ...

- Assess their quality

- Make their use and processing a central element of business activities

  Including quality assessment and cleaning

Only quality data lead us to the right conclusions and decisions

There has been a lot of research on data cleaning

There are many *ad hoc* solutions that are rigidly vertical:

- Applicable to specific problems and domains
- Difficult to reuse or adapt

  Even to slightly different scenarios

There are many tools without a theory that allows us to understand what they (do not) do

We do not fully understand yet the basic foundations of data quality assessment and data cleaning

We have missed the big picture and the general principles that underly data quality assessment and cleaning

Things are starting to change ...

Generic Data Cleaning

More recent research thrust is about:

- Developing data quality solutions that have a broader scope of applicability
- Proposing general, flexible and parameterizable solutions

  Easily adaptable according to the specific problem and domain at hand and need
- Providing declarative solutions that can be easily understood in terms of what they mean and do

  Instead of being hardwired in complex purely algorithmic solutions and programs
- Providing conceptual and mathematical frameworks on top of which data cleaning activities can be solidly built and evaluated

A Declarative Approach to ER

A generic way to approach the problem consists in specifying attribute values that have to be matched (made identical) under certain conditions

A declarative language with a precise semantics could be used for this purpose

In this direction, matching dependencies (MDs) were recently introduced (Fan et al., PODS'08, VLDB'09)

They represent rules for resolving pairs of duplicate representations (two tuples at a time)

An MD indicates attribute values that have to be matched when certain similarities between attribute values hold

Example: The similarities of phone and address indicate that the tuples refer to the same person, and the names should be matched

People (P)	Name	Phone	Address
	John Smith	723-9583	10-43 Oak St.
	J. Smith	(750) 723-9583	43 Oak St. Ap. 10

Here: 723-9583 $\approx$ (750) 723-9583 and 10-43 Oak St. $\approx$ 43 Oak St. Ap. 10

An MD capturing this cleaning policy:

$$P[Phone] \approx P[Phone] \wedge P[Address] \approx P[Address] \ \rightarrow$$
$$P[Name] \doteq P[Name]$$

(an MD may involve two different relations)

Matching Dependencies

MDs are rules of the form

$$\bigwedge_{i,j} R[A_i] \approx_{ij} S[B_j] \ \rightarrow \ \bigwedge_{k,l} R[A_k] \doteq S[B_l]$$

LHS captures a similarity condition on pairs of tuples, in relations R and S

Abbreviation: $R[\bar{A}] \approx S[\bar{B}] \ \rightarrow \ R[\bar{C}] \doteq S[\bar{E}]$

$\approx$: Domain-dependent similarity relations

Can be specified and imposed

Dynamic interpretation: Those values on the RHS should be updated to some common value

What semantics to assign to MDs and the process of applying them to a DB?

Although declarative, MDs have a procedural feel and a dynamic semantics

An MD (or set thereof) 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

We have to iteratively enforce the MDs until ER is solved (a chase procedure)

$$D \mapsto D_1 \mapsto D_2 \mapsto \cdots \mapsto D^c \qquad \text{(clean)}$$

$\mapsto$: How?

MDs as originally introduced do not say how to identify values

$$\forall X_1 X_2 Y_1 Y_2 (R_1[X_1] \approx R_2[X_2] \quad \longrightarrow \quad R_1[Y_1] \doteq R_2[Y_2])$$

We have considered the two (families of) operational semantics:

- With matching functions (MFs)

  (Bahmani, Bertossi, Kolahi, Lakshmanan)

  A domain-dependent, underlying function tells us which value to pick up for a value in common

- Without MFs (Gardezi, Bertossi, Kiringa)

  A value for a matching is arbitrarily chosen, but the final result (clean instance) has to minimize the number of attribute changes

Possibly several "resolved" instances may be obtained

Example: (with MFs)

"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

A dynamic semantics!

$$m_{address}(\underline{MainSt., Ottawa}\ ,\ \underline{25MainSt.}) := \underline{25MainSt., Ottawa}$$

Example: Assume only these two resolved instances for D:

D_1^c	*name*	*address*
	John Doe	25 Main St., Ottawa
	J. Doe	25 Main St., Ottawa
	Jane Doe	25 Main St., Vancouver

D_2^c	*name*	*address*
	John Doe	Main St., Ottawa
	J. Doe	25 Main St., Vancouver
	Jane Doe	25 Main St., Vancouver

(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 $\mathcal{Q}$: `SELECT * FROM R`

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

- $\mathit{Certain}(\mathcal{Q}, D) = \{\langle$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

Computation/materialization of *all* resolved instances is undesirable!

- Query rewriting: Transform query $\mathcal{Q}$ into new, tractable query $\mathcal{Q}'$ to be posed only and as usual to original instance D
- Declaratively, logically specify the whole class of resolved instances using *answer set programs* (ASP)

  Resolved instances are implicit

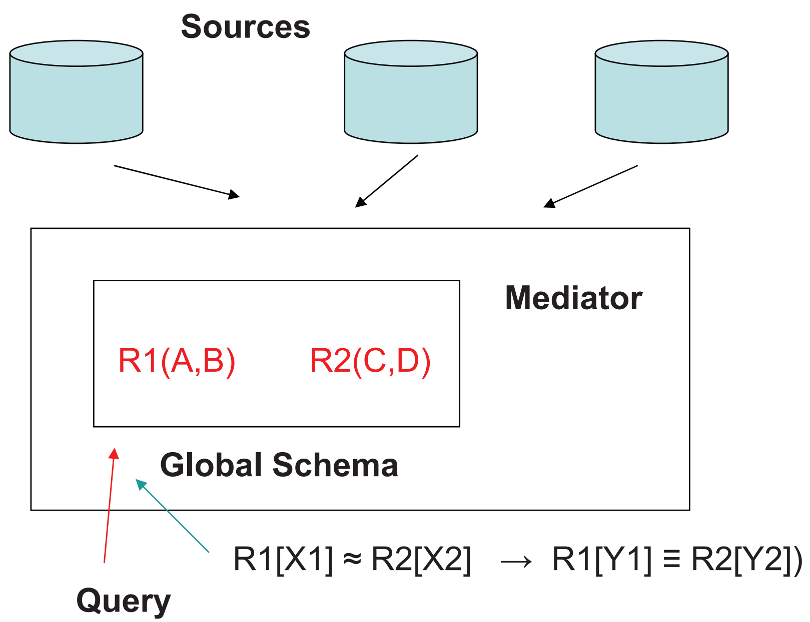
  Skeptically reason with (query) the ASP to obtain certain answers

Ongoing Research

Declarative specifications for ER can be compiled into query answering!

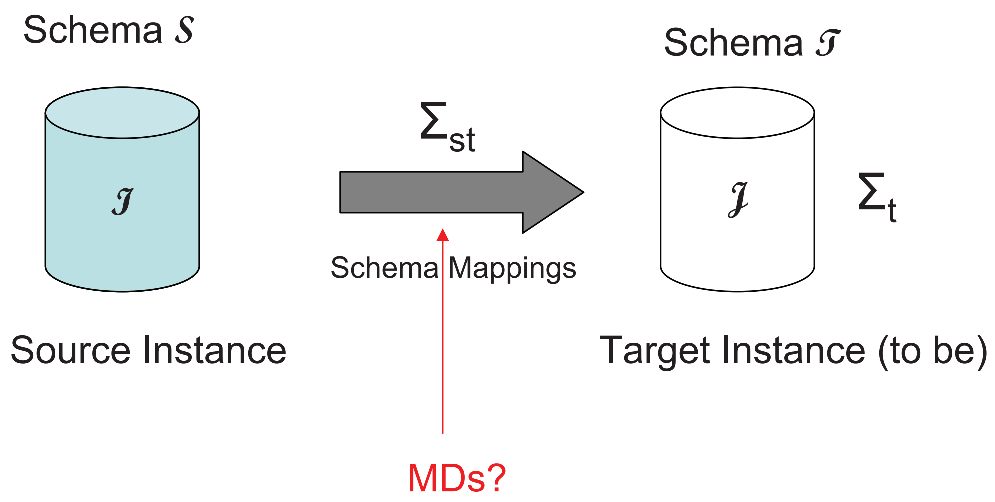
For different applications

Virtual data integration: A natural application scenario



On-the-fly ER!

Data exchange under schema mappings:



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

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

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

ER at data exchange time ... (Bahmani, Bertossi, Geerts)

MDs and uncertain data:

Above:

- We know about similarity relations
- We know the MDs

Data is more uncertain ...

There is some work on discovering MDs, a data mining task

We can go beyond and propose and apply uncertain MDs

At the same time inferring and using uncertain similarity relations

Example: $R_1(A,B),\ R_2(C,D)$ $\quad$ $Dom(A) = Dom(C) = Do_1$

$$Dom(A) = Dom(C) = Do_2$$

MD: $\quad R_1[X_1] \approx_1 R_2[X_2] \ \rightarrow\ R_1[Y_1] \doteq R_2[Y_2]$

- $\doteq$ is equality after matching, a particular form of similarity

So, possibly: $\quad R_1[X_1] \approx_1 R_2[X_2] \ \rightarrow\ R_1[Y_1] \dot{\approx}_2 R_2[Y_2]$

- MDs may have certain probabilities (or weights) assigned

0.8 : $People[Phone] \approx_{ph} People[Phone] \ \wedge People[Address] \approx_{ad} People[Address]$
$$\longrightarrow \quad People[Name] \doteq People[Name]$$

- Or a finer granularity:

$People[Phone] \approx_{ph}^{\mathbf{0.8}} People[Phone] \wedge People[Address] \approx_{ad}^{\mathbf{0.5}} People[Address]$
$$\rightarrow People[Name] \doteq^{\mathbf{0.7}} People[Name]$$

General issue: Chase-like enforcement of MDs with probability propagation

Possibly similar to chase for probabilistic Datalog$\pm$

(Gottlob et al.)

In combination with Markov logic networks (MLNs)

(Domingos et al.)

In our case, an approach based on the combination of MDs and MLNs would allow for both learning and probabilistic reasoning

Many issues are being investigated ...