Introduction

- Data is inherently heterogeneous
  - Due to the explosion of online data repositories
  - Due to the variety of users, who develop a wealth of applications
    - At different time
    - With disparate requirements in their mind

- A fundamental requirement is to translate data across different formats
  - How data is transformed from one format to another is done through mappings
Mappings Are All Around

- Data integration [Lenzerini 2002]
  - to specify the relationship between local and global schemas
Mappings Are All Around

- Schema integration [Batini et al. 1986]
  - to specify the relationship between the input schemas and the integrated schemas
Data exchange [Fagin et al. 2005] to specify the relationship between source and target schemas.
Mappings Are All Around

- Schema evolution [Lerner 2000]
  - to specify the relationship between the old and new version of an evolved schema

Evolving Schema S1

mappings
One of the first systems to deal with this problem was developed at IBM in 1977: EXPRESS (EXtraction, Processing and REStructuring System) [Shu et al. 1977] consists of two languages:

- DEFINE that works as a DDL (Data Definition Language)
- CONVERT that works as a DTL (Data Translation Language) and has a total of 9 operators, each of which receives as input a data file, performs the respective transformation and generates an output data file.

EXPRESS required the users familiarity with the languages and was customized to only one model (hierarchical)

After that, inter-model transformations were also studied [Tork-Roth et al. 1997] [Atzeni et al. 1997]
A Data Transfer Example

Source Instance

<table>
<thead>
<tr>
<th>Projects</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIX</td>
<td>Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-services</td>
<td>Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clio</td>
<td>Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grants</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>project</td>
<td>recipient</td>
<td>manager</td>
<td>supervisor</td>
</tr>
<tr>
<td>g1</td>
<td>PIX</td>
<td>AT&amp;T</td>
<td>Fernandez</td>
<td>Belanger</td>
</tr>
<tr>
<td>g2</td>
<td>PIX</td>
<td>AT&amp;T</td>
<td>Shrivastava</td>
<td>Belanger</td>
</tr>
<tr>
<td>g3</td>
<td>E-services</td>
<td>Bell-labs</td>
<td>Benedikt</td>
<td>Hull</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contacts</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cid</td>
<td>email</td>
<td>phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benedikt</td>
<td><a href="mailto:benedikt@research.bell-labs.com">benedikt@research.bell-labs.com</a></td>
<td>5827766</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hull</td>
<td><a href="mailto:hull@research.bell-labs.com">hull@research.bell-labs.com</a></td>
<td>5824509</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shrivastava</td>
<td><a href="mailto:diversh@research.att.com">diversh@research.att.com</a></td>
<td>3608776</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belanger</td>
<td><a href="mailto:dbg@research.att.com">dbg@research.att.com</a></td>
<td>3608600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fernandez</td>
<td><a href="mailto:mff@research.att.com">mff@research.att.com</a></td>
<td>3608679</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Companies</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>official</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>AT&amp;T Research Labs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lucent</td>
<td>Lucent Technologies, Bell Labs Innovations</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Target Instance

<table>
<thead>
<tr>
<th>Company</th>
<th>name</th>
<th>official</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>AT&amp;T</td>
<td>Research Labs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lucent</td>
<td>Lucent</td>
<td>Technologies, Bell Labs Innovations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Desired Target Instance

Source: `Rcd`
- `projects`: Set of `Rcd`
  - `name`
  - `status`
- `grants`: Set of `Rcd`
  - `gid`
  - `project`
  - `recipient`
  - `manager`
  - `supervisor`
- `contacts`: Set of `Rcd`
  - `cid`
  - `email`
  - `phone`
- `companies`: Set of `Rcd`
  - `coid`
  - `name`
  - `official`

Target: `Rcd`
- `projects`: Set of `Rcd`
  - `code`
- `funds`: Set of `Rcd`
  - `fid`
  - `finId`
  - `mPhone`
  - `company`
- `finances`: Set of `Rcd`
  - `finId`
- `companies`: Set of `Rcd`
  - `coid`
  - `name`

### Projects

<table>
<thead>
<tr>
<th>code</th>
<th>Funds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>fid</code></td>
</tr>
<tr>
<td>g3</td>
<td>???</td>
</tr>
</tbody>
</table>

### Finances

<table>
<thead>
<tr>
<th>finId</th>
<th>mPhone</th>
<th>company</th>
</tr>
</thead>
<tbody>
<tr>
<td>???</td>
<td>3608679</td>
<td>???</td>
</tr>
<tr>
<td>???</td>
<td>3608776</td>
<td>???</td>
</tr>
<tr>
<td>???</td>
<td>5827766</td>
<td>???</td>
</tr>
</tbody>
</table>

### Companies

<table>
<thead>
<tr>
<th>coid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sk2(AT&amp;T)</td>
<td>AT&amp;T</td>
</tr>
<tr>
<td>Sk2(Lucent)</td>
<td>Lucent</td>
</tr>
<tr>
<td>???</td>
<td>???</td>
</tr>
<tr>
<td>???</td>
<td>???</td>
</tr>
</tbody>
</table>

The Needed Transformation Query

LET $doc0 := document("inputXMLfile")
RETURN
<T> { distinct-values (
  FOR $x0 IN $doc0/S/grant, $x1 IN $doc0/S/project, $x2 IN $doc0/S/contact, $x3 IN $doc0/S/contact
  WHERE $x2/cid/text() = $x0/manager/text() AND $x0/supervisor/text() = $x3/cid/text() AND $x0/project/text() = $x1/name/text()
  RETURN
    <project>
      <code> { $x0/project/text() } </code>
    </project>
  )
) } { distinct-values (
  FOR $x0L1 IN $doc0/S/grant, $x1L1 IN $doc0/S/project, $x2L1 IN $doc0/S/contact, $x3L1 IN $doc0/S/contact
  WHERE $x2L1/cid/text() = $x0L1/manager/text() AND $x0L1/supervisor/text() = $x3L1/cid/text() AND $x0L1/project/text() = $x1L1/name/text() AND $x0/project/text() = $x0L1/project/text()
  RETURN
    <funding>
      <fid> { $x0L1/gid/text() } </fid>
      <finId> { "Sk52(" , $x0L1/gid/text(), ", ", $x0L1/project/text(), ")" } </finId>
    </funding>
  )
) }

RETURN
<finance> { $x0/gid/text() } </finance>
<mPhone> { $x2/phone/text() } </mPhone>
<company> { "Sk46(" , $x2/phone/text(), ", ", $x0/gid/text(), ")" } </company>
</finance>

{ distinct-values (
  FOR $x0 IN $doc0/S/company
  RETURN
    <company>
      <coid> { "Sk46(" , $x0/cname/text(), ")" } </coid>
      <name> { "Sk49(" , $x2/phone/text(), ", ", $x0/gid/text(), ")" } </name>
    </company>
  )
) } { distinct-values (
  FOR $x0 IN $doc0/S/company
  RETURN
    <company>
      <coid> { "Sk93(" , $x0/cname/text(), ")" } </coid>
      <name> { $x0/cname/text() } </name>
    </company>
  )
) }

© Journées LSC du LIP 2016, A. Bonifati
The design of data transformations has been a manual task for a long while

- Designers had to be familiar with the language
- As schemas became larger and more complex, the task became too laborious, time-consuming and error-prone

The need of raising the level of abstraction and trying to automate the tasks was soon realized.

The idea ...

© Journées LSC du LIP 2016, A. Bonifati
Different techniques exist to generate mappings:

- Manual, e.g.
  - by means of high-level mapping languages, such as [Bernstein et al. 2007]
  - by means of sophisticated user interfaces [Altova 2008]

- Semi-automatic, e.g.
  - By means of designer guidance [Alexe 2008]
  - Via advanced algorithms to do the reasoning instead of the mapping designer [Madhavan at al. 2001] [Popa et al. 2002] [Do et al. 2002] [Bonifati et al. 2008]
The First Step of a Mapping Task

<table>
<thead>
<tr>
<th>Projects</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>status</td>
<td></td>
</tr>
<tr>
<td>PIX</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>E-services</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>Clio</td>
<td>Inactive</td>
<td></td>
</tr>
</tbody>
</table>

| Grants | | |
| --- | --- | --- | --- |
| gid | project | recipient | manager | supervisor |
| g1 | PIX | AT&T | Fernandez | Belanger |
| g2 | PIX | AT&T | Shrivastava | Belanger |
| g3 | E-services | Bell-labs | Benedikt | Hull |

<table>
<thead>
<tr>
<th>Contacts</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cid</td>
<td>email</td>
<td>phone</td>
</tr>
<tr>
<td>Benedikt</td>
<td><a href="mailto:benedikt@research.bell-labs.com">benedikt@research.bell-labs.com</a></td>
<td>5827766</td>
</tr>
<tr>
<td>Hull</td>
<td><a href="mailto:hull@research.bell-labs.com">hull@research.bell-labs.com</a></td>
<td>5824509</td>
</tr>
<tr>
<td>Shrivastava</td>
<td><a href="mailto:divesh@research.att.com">divesh@research.att.com</a></td>
<td>3608776</td>
</tr>
<tr>
<td>Belanger</td>
<td><a href="mailto:dgb@research.att.com">dgb@research.att.com</a></td>
<td>3608600</td>
</tr>
<tr>
<td>Fernandez</td>
<td><a href="mailto:mff@research.att.com">mff@research.att.com</a></td>
<td>3608679</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Companies</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>official</td>
<td></td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>AT&amp;T Research Labs</td>
<td></td>
</tr>
<tr>
<td>Lucent</td>
<td>Lucent Technologies, Bell Labs Innovations</td>
<td></td>
</tr>
</tbody>
</table>
Matching

- Given two schemas as input, the source and the target schema, matching is a process that produces as output a set of matches or correspondences or (simply) lines between the elements of the two schemas.

- A match is a triple $<E_s, E_t, e>$
  - Where $E_s$ is a set of elements of the source schema, $E_t$ is a set of elements of the target schema, and $e$ specifies a simple relationship (equality or set inclusion) or complex relationship between element in $E_s$ and $E_t$.
The Matching Relationship $e$

- Depends on the cardinalities of $E_s$ and $E_t$
- Depends on the semantics:
  - Can be a function
  - Can be an arithmetic operation
  - Can be a set-theoretic relation (e.g. $\equiv$, overlaps)
  - Can be a conceptual modeling relationship (e.g. part-of, subclass-of)
Matching: An Alternative Definition

- The matching process [Euzenat et al. 2007] can be seen as a function $f$ from a pair of schemas $S$ and $T$, an optional input alignment $A$, a set of matching parameters $p$ and a set of resources $r$:

$$A' = f(S, T, A?, p, r)$$

- Ultimately, an alignment is a set of correspondences between elements in $S$ and elements in $T$.  

Matching Examples

- **Simple relationship:**
  - Name $\equiv$ Title
  - Location $\equiv$ Address

- **Complex relationship:**
  - speed = velocity $\times$ 2.237
  - speed $\times$ 0.447 = velocity
  - speed = concat(velocity $\times$ 2.237, ‘MPH’)
  - speed $\geq$ velocity
The matching process

- Can be roughly divided into three steps:
  - Pre-match: training of classifiers for machine learning-based matchers, matching parameters (weights, thresholds), adjustments of resources, such as thesauri and constraints
  - Match: the actual matching task
  - Post-match: the user may check and modify the displayed matches
Some Schema Matchers

- Cupid [Madhavan et al. 2001]: based on structural and name similarity
- S-Match [Giunchiglia et al. 2004]: based on semantic closeness
- Coma++ [Aumueller et al. 2005]: based on matching reuse
- LSD [Doan et al. 2001]: based on data value analysis and machine-learning techniques
- iMap [Dhamankar et al. 2004]: suited for complex expressions
- Similarity Flooding [Melnik et al. 2002]: based on graph similarity
Similarity Flooding

```xml
<xml version="1.0">  
  <element name="searchform">  
    <element name="fid"/>  
    <element name="Hotel Location">  
      <element name="* City"/>  
      <element name="* Zip Postal Code"/>  
      <element name="* State"/>  
      <element name="Country:"/>  
    </element>  
    <element name="Reservation Details">  
      <element name="Check-in date day"/>  
      <element name="Check-in date month"/>  
      <element name="Check-in date year:"/>  
      <element name="Check-out date day"/>  
      <element name="Check-out date month"/>  
      <element name="Check-out date year:"/>  
      <element name="Number of Adults:"/>  
      <element name="Children:"/>  
      <element name="Rooms Needed:"/>  
    </element>  
  </element>  
  <element name="Optional">  
    <element name="Hotel Brand:"/>  
  </element>  
</xml>  
</xml>
```
COMA++
Matchings Are Not Enough

Projects

<table>
<thead>
<tr>
<th>name</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIX</td>
<td>Active</td>
</tr>
<tr>
<td>E-services</td>
<td>Active</td>
</tr>
<tr>
<td>Clio</td>
<td>Inactive</td>
</tr>
</tbody>
</table>

Grants

<table>
<thead>
<tr>
<th>gid</th>
<th>project</th>
<th>recipient</th>
<th>manager</th>
<th>supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>PIX</td>
<td>AT&amp;T</td>
<td>Fernandez</td>
<td>Belanger</td>
</tr>
<tr>
<td>g2</td>
<td>PIX</td>
<td>AT&amp;T</td>
<td>Shrivastava</td>
<td>Belanger</td>
</tr>
<tr>
<td>g3</td>
<td>E-services</td>
<td>Bell-labs</td>
<td>Benedikt</td>
<td>Hull</td>
</tr>
</tbody>
</table>

Contacts

<table>
<thead>
<tr>
<th>cid</th>
<th>email</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benedikt</td>
<td><a href="mailto:benedikt@research.bell-labs.com">benedikt@research.bell-labs.com</a></td>
<td>5827766</td>
</tr>
<tr>
<td>Hull</td>
<td><a href="mailto:hull@research.bell-labs.com">hull@research.bell-labs.com</a></td>
<td>5824509</td>
</tr>
<tr>
<td>Shrivastava</td>
<td><a href="mailto:divesh@research.att.com">divesh@research.att.com</a></td>
<td>3608776</td>
</tr>
<tr>
<td>Belanger</td>
<td><a href="mailto:dgb@research.att.com">dgb@research.att.com</a></td>
<td>3608600</td>
</tr>
<tr>
<td>Fernandez</td>
<td><a href="mailto:mff@research.att.com">mff@research.att.com</a></td>
<td>3608679</td>
</tr>
</tbody>
</table>

Companies

<table>
<thead>
<tr>
<th>name</th>
<th>official</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>AT&amp;T Research Labs</td>
</tr>
<tr>
<td>Lucent</td>
<td>Lucent Technologies, Bell Labs Innovations</td>
</tr>
</tbody>
</table>

Cannot describe the full details of the transformation
The Mapping Generation Process

Source Schema → Matcher → Target Schema

Matchings

Matching is just the beginning of any mapping generation process
Mappings

- Given the source and the target schemas, mapping is a process that takes as input a set of matches between the elements of the two schemas and produces a relationship or constraint $e$ that must hold between their respective instances.

- In other words, a mapping is a triple $<S, T, e>$
  - Where $S$ is the source schema, $T$ is the target schema, and $e$ specify a constraint that any instances adhering to $S$ and $T$ must satisfy or an executable statement to transform the instance of $S$ into the instances of $T$. 
A Mapping Example

\[
\text{project}(\text{na, st}), \quad \text{grant}(\text{gid, na, re, ma, su}), \quad \text{contact}(\text{ma, em, ph}) \quad \rightarrow \\
\text{project}(\text{na, FUND}), \quad \text{fund}(\text{gid, finId}), \quad \text{finance} \quad (\text{finId, ph, company}) \\
\text{company}(\text{company, name})
\]
Mappings & Instances

- Mappings are the basic ingredients of many tasks, such as information integration, P2P query answering, data exchange etc.

- In particular, mappings as inter-schema constraints may not be enough to fully specify a unique target instance
  - There may exist multiple target instances satisfying the mappings
  - Finding the best target instance is the goal of the data exchange problem [Fagin et al. 2005]
    - The mapping is converted into an executable transformation script to obtain that particular instance
A Data Exchange Example

S: Rcd
  projects: Set of project: Rcd
    name
    status
  grants: Set of grant: Rcd
    gid
    project
    recipient
    manager
    supervisor
  contacts: Set of contact: Rcd
    cid
    email
    phone
  companies: Set of company: Rcd
    coid
    name
    official

T: Rcd
  projects: Set of project: Rcd
    code
    funds: Set of fund: Rcd
      fid
      finId
  finances: Set of finance: Rcd
    finId
    mPhone
    company
  companies: Set of company: Rcd
    coid
    name

projects: Set of project: Rcd

A Data Exchange Example

Target instance

Projects
  code: E-services
  Funds
    fid | finId
    g3 | ???

code: PIX
  Funds
    fid | finId
    g1 | ???
    g2 | ???

Finances
  finId | mPhone | company
  ???  | 3608679 | ???
  ???  | 3608776 | ???
  ???  | 3608600 | ???

Companies
  coid | name
  ???  | AT&T
  ???  | Lucent

project(\text{na, st}), \text{grant}(\text{gid, na, re, ma, su}), \text{contact}(\text{ma, em, ph}) \rightarrow
project(\text{na, FUND}), \text{fund}(\text{gid, finId}), \text{finance}(\text{finId, ph, company}),
\text{company}(\text{company, name})
The Mapping Generation Process

Source Schema

Matcher

Matchings

Mapping Generation Engine

Mappings (Dependencies)

Data Exchange Engine

Transformation Scripts

Query Engine

Target Instance

Source Instance

© Journées LSC du LIP 2016, A. Bonifati
Mapping generation and data exchange are separate tasks
- Clio[Popa et al. 2002], HePToX[Bonifati et al. 2010], Spicy[Bonifati et al. 2008]

Mappings Generation
- the mappings are expressed as high-level assertions in a logical formalism
- A mapping is a source-to-target tuple-generating dependency (or s-t tgd in short)
  \[ \phi(x) \rightarrow \exists \psi(x, y) \] where
- \( \phi(x) \) (\( \psi(x, y) \), resp.) is a conjunction of atoms over the source (target, resp.)

Data exchange
- The respective module transforms the high-level mappings into transformation scripts (in SQL or XQuery) to generate the target instance.
HepToX
Commercial Mapping Systems

Mapping generation and Data exchange are merged in one. The system directly creates in some native language the final transformation script.
Popular Commercial Systems

- Altova Mapforce
- Stylus Studio
- IBM Rational Data Architect
- BizTalk mapper
- Adeptia
- BEA Aqualogic
Altova MapForce
BizTalk Mapper

Source Specification:
- NorthwindReq
  - Header
    - reqNumber
    - reqStatus
    - dateCreated
    - timeCreated
  - comments
  - Shipping
    - name
    - addr1
    - addr2
    - city
    - state
    - zip
    - country
    - phone

Destination Specification:
- ContosoPO
  - POHeader
    - referenceNumber
    - created
    - ShipTo
      - name
      - address
      - city
      - state
      - zip
      - country
      - phone
  - Item
    - partNo
    - description

Properties:
- Name: Northwind
- Description

Values:
- Value

Warnings:

© Journées LSC du LIP 2016, A. Bonifati
A Mapping Tool Categorization

- All tools provide to the mapping designer:
  - A graphical representation of the two schemas
  - A set of graphical transformation constructs

  - The granularity and power of these constructs is a main factor of differentiation among the tools

Intelligence of the mapping tool/
Effort in post-verification

- Roughly Research Prototypes
- Roughly Commercial Mapping Tools

Detailed Specification by the Designer
Issues in Data Exchange

- When multiple target instances exist, how do we compute the best one?
- Is a given target instance better than another?
  - Universal solutions
    - Introduced in [Fagin et al. 2005]
    - These are the “most general” target instances, and also represent the entire space of solutions
    - Among the universal solutions, the smallest of all and the most compact one is called the “core”
Universal Core Instances

Source instance

**PTStud**

<table>
<thead>
<tr>
<th>age</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>John</td>
</tr>
<tr>
<td>30</td>
<td>Ann</td>
</tr>
</tbody>
</table>

**GradStud**

<table>
<thead>
<tr>
<th>age</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Bob</td>
</tr>
<tr>
<td>30</td>
<td>Ann</td>
</tr>
</tbody>
</table>

**PTStud** (x,y), \(\rightarrow\) **Advised** (y,z)

**GradStud** (x,y) \(\rightarrow\) **Advised** (y,z), **WorksWith** (y,z)

A universal Solution:

**Advised**

<table>
<thead>
<tr>
<th>sname</th>
<th>facid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
<tr>
<td>John</td>
<td>N1</td>
</tr>
<tr>
<td>Cathy</td>
<td>N1</td>
</tr>
</tbody>
</table>

**WorksWith**

<table>
<thead>
<tr>
<th>sname</th>
<th>facid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
<tr>
<td>Cathy</td>
<td>N1</td>
</tr>
</tbody>
</table>

A solution:

<table>
<thead>
<tr>
<th>sname</th>
<th>facid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
</tbody>
</table>

Possible instances

The core:
Commercial vs. Research Prototype Systems

- Whereas research prototypes (e.g. Clio, Spicy++) are tending to produce target instances that look more and more like the core.

- Commercial tools leave the task to the users, who have to manually interact with sophisticated GUIs and write pieces of the transformation manually.
  - No core definition is even considered.

Core? No, thanks
Matching and Mapping not always by separate tools

- Clio has as an add-on a matcher based on attribute feature analysis [Naumann et al. 2002]
- Bernstein’s model management considers the matcher as a fully integrated and indistinguishable component
- Spicy [Bonifati et al. 2008] has a matcher based on instance-based structural analysis
Limitations of Current Systems

- Manual approaches are not applicable to large-scale mapping tasks
- The user/developer has to become familiar with the mapping language and the user interfaces
- The outcome of the mapping process may not respect the user requirements and desired semantics (unsurprisingly!)
- Specifications may be incomplete and dependent of system peculiarities
- Thus, there is a need for a verification and guidance process
The Verification Process

Source Schema

- Matcher
- Matchings
- Mapping Generation Engine
- Mappings (Dependencies)
- Data Exchange Engine

<table>
<thead>
<tr>
<th>Source Instance</th>
<th>Query Engine</th>
<th>Target Instance</th>
</tr>
</thead>
</table>

Target Schema

- User
- Verification And Selection
- Expected Target Instance

Data Examples

User

Data Exchange Engine

Transformation Scripts

Mapping Generation Engine

Matching

Mapping Generation Engine

Data Exchange Engine
The main problem with matching and mapping is the dichotomy between the expected results and the generated answers.

Some tools allow a post-verification:
- by using data examples:
  - Tupelo [Fletcher et al. 2006], Muse [Alexe et al. 2008], Clio [Alexe et al. 2010]
- by using an automatic instance comparison:
  - Spicy [Bonifati et al. 2008]
- by means of manual user feedback:
  - unfeasible for large-scale tasks
- via debugging techniques:
  - Routes [Chiticariu et al. 2006]
Conclusions and Ongoing work

- Interactive Mapping Refinement with User Data Examples
  - Independence of the graphical language
  - Approach driven by data examples that are simple to build for a generic user (few tuples/no core or universal data examples)
  - Obtained mappings have good properties (normalisation, absence of redundancy)
  - Joint work with U. Comignani (M2 Lyon1), E. Coquery and R. Thion


Thank you!!

Thanks to my co-author Y. Velegrakis (Univ. of Trento, Italy)
• [Lenzerini 2002] Lenzerini Maurizio “Data Integration: A Theoretical Perspective”. In: PODS, pp 233-246
List of References (cont’d)

List of References (cont’d)