ODDI
a Framework for Semi-automatic Data Integration

Paolo Ceravolo, Ernesto Damiani, Marcello Leida
Università degli studi di Milano, Dipartimento di Tecnologie dell’Informazione, via Bramante, 65, 26013 Crema (CR), Italy
ceravolo@dti.unimi.it, damiani@dti.unimi.it, leida@dti.unimi.it

Zhan Cui, Alex Gusmini
Intelligent Systems Research Centre, BT Group, Orion Building - Adastral Park - Martlesham Heath, IP5 3RE Ipswich - Suffolk, UK
zhan.cui@bt.com, alex.gusmini@bt.com

Keywords: Data Integration; Mapping Generation; Matching Operators; Clustering; Semantic; Formal Concept Analysis

Abstract: Recent works on Business Intelligence do highlight the need of on-time, trustable and sound data access systems. Moreover the application of these systems in a flexible and dynamic environment requires for an approach based on automatic procedures that can provide reliable results. A crucial factor for any automatic data integration system is the matching process. Different categories of matching operators carry different semantics. For this reason combining them in a single algorithm is a non trivial process that have to take into account a variety of options. This paper proposes a solution based on a categorization of matching operators that allow to group similar attributes on a semantic rich form. This way we define all the information need in order to create a mapping. Then Mapping Generation is activated only on those set of elements that can be queried without violating any integrity constraints on data.

1 Introduction

Data Integration is becoming a relevant problem in applications that needs to access, analyse and display data coming from heterogeneous data sources.

In principle Data Integration can be done by a procedural approach, i.e. creating an ad-hoc integration with respect to a set of predefined needs, such as in (Hammer et al., 1995). But, when the queries to be applied on the sources cannot be define a-priori, a declarative approach is required. Due to flexibility requirements the declarative approach is more diffused, and here we limit our discussion to it. According to the declarative approach, we call local schemata \( L \) the set of representations referring to local sources, while the global schema \( G \) is the representation integrating the different local sources. Research issues regarding Data Integration problem can be grouped in three big clusters: a first cluster of issues focus on the generation of \( G \) that can be either normative, as in (Braun et al., 2000), or inductive, as in (Hakimpour and Geppert, 2002). A second cluster of issues focus on how to represent the mapping between \( G \) and \( L \). Here two main approaches exist. In the Global as View approach the mapping is provided on \( G \) objects by using a \( L \) vocabulary while in the Local as View approach the mapping is provided on \( L \) objects by using a \( G \) vocabulary. In (Lenzerini, 2002) a detailed discussion underlines how these approaches impact on application modeling and data reasoning. The last cluster of issues focuses on the problem of query answering, studying the computational complexity related to the different solutions, as in (Abiteboul and Duschka, 1998) or in (Halevy, 2001), and defining effective algorithms for dealing to it, as for instance in (Grahne and Mendelzon, 1999) or (Duschka et al., 2000). These problems cover nearly the totality of the relevant theoretical aspects involved in Data Integration. Moreover, with the increasing number of interaction and complexity of relations, human interaction is becoming an help that can not be considered anymore: the mapping must be generated by means of an automatism providing high level of quality in the final mapping between the data sources analyzed. Here outcomes the importance of sound matching operators that can discover semantics relations between elements of the system.

Matching operators are very susceptible to the
data in input, because different operators are tailored to different data, and no generic matching function can be designed. For this reason the only way for implementing a generic Data Integration algorithm is to support different matching operators. Moreover, combining them in a single algorithm is a non trivial process that have to take into account a variety of options.

This paper deals with the problem of managing a pallet of matching operator supporting different semantics. The approach chosen is to combine all the available association produced by different operators in a cluster. This cluster collect all the elements that can be associated and express the semantics of the associations. This way in the cluster we have all the available association produced by different operators: relations between objects of the local representation L and the global representation G (such as for instance concepts in an ontology or table in a database) and M is a mapping between elements of L: relations between objects of the same source schema s_i in L, such as the typical primary-key—foreign-key, but also relations between elements of different source schemas s_i, s_j of L that are semantically related.

In general two data set can be integrated only if they describe a common set of real world facts. Of course this common set does not have to cover the totality of the described facts. In (Parent and Spaccapietra, 1998), relations between facts described by different data sets are defined by set relationships. Actually this approach is partially inappropriate because the instances of two data sets can describe the same facts at different detail levels or they can describe distinct facts to be related in G. According to our work a mapping between data sets can be oriented to two distinct goals:

- **Composition.** In this case some redundant information is assumed to be stored in the data sets. The mapping acts on this redundant information in order to aggregate new compositions of data items. In this perspective G contains views that recompose the data items contained in L in a new structure.

- **Summarization.** In this case the information stored in the data sets can be reduced to a common type. The mapping expresses the communality shared by different data items. In this perspective G contains views that summarize the data items contained in L in a more compact representation.

In principle a mapping can cover both these goals. If an human agent generate the mapping, she will naturally distinguish between the two cases. But if the mapping is generated by an algorithm, achieving the right goal mainly depends on the operator adopted for matching the data items.

The system that we propose consists of two modules: the first that generates M_\_p, given G and L. The second module generates M_o, by representing M_\_p as an FCA used as searching space to find semantic relations between elements of G and L.

### 2 Data Integration

The system we propose in this paper is based on Global as View approach (Calvanese et al., 1998), because the G is given trough an ontology and the mapping are constructed by associating to the concepts of G the set of attributes in L that carry the same informative value of the attributes of these concepts. Formally we can define a data integration system as triple I =< G, L, M >; where G is the global representation, L the local set of local representations composed by n single representations s_1, s_2, ..., s_n and M is the mapping between L and G. The mapping M is the result of a complex process taking as input M_\_p, a set of matching relations among the simple elements of G and L and generating the mapping M defined as M =< M_\_p, M_\_o >; where M_\_p is a mapping between objects of the local representation L and the global representation G (such as for instance concepts in an ontology or table in a database) and M_\_o is a mapping between elements of L: relations between objects of the same source schema s_i in L, such as the typical primary-key—foreign-key, but also relations between elements of different source schemas s_i, s_j of L that are semantically related.
3 Matching

Matching ($M_i$) is the problem of discovering relations between elements of two different representations ($G$ and $L$ in this case). The matching at simple element level can be defined as a relation:

$$e_i^s \cong \delta e_i^l$$

Where $\cong$ is a binary relation from the set of the following relations: equality, inclusion and specification ($\equiv$, $\subset$, $\subset^\circ$). While $\delta$, associated to $\cong$, can be a binary $\{0,1\}$ or a fuzzy value $\{0..1\}$ depending on the method chosen to implement the matching operator and represents the strength of the relation. Then elements of $m_i$ are: $e_i^s = e_i^l$ if the data items carried by $e_i^s$ satisfy $e_i^l$ with a sound and complete information (i.e. name and firstName). Or in case of inclusion $e_i^s < e_i^l$, if the data items carried by $e_i^s$ provide a portion of the information required for satisfying $e_i^l$ (i.e. street and address). $e_i^s < e_i^l$ if the data items carried by $e_i^s$ contain more information than the one required for satisfying $e_i^l$ (i.e. street and address). Or, in case of specification, $e_i^s \subset e_i^l$ if the data items carried by $e_i^s$ satisfy $e_i^l$ with a sound but more general information (i.e. email and contact).

In addition an element $e$ can also be subject of a transformation, allowing to bring $e$ under the range of a binary relation. We use $\Delta$ as a symbol for meaning these transformation. So we can have a matching such as:

$$\Delta e_i^s \cong \delta e_i^l$$

if the information carried by $e_i^s$ allow a transformation that satisfy the semantics of $e_i^l$. As an example, if $e_i^s$ represent the nationality of a person and $e_i^l$ his telephone number, we can derive, using a pattern based operator, the nationality from the international prefix of the telephone number.

The values $\cong$ and $\delta$ are the output of a matching function $\phi(e_i^s, e_i^l)$ that evaluates the semantic relations between two elements $e_i^s$ and $e_i^l$.

A variety of methods for implementing the matching function $\phi$ were proposed in the literature in order to define correspondences among elements belonging to different representations. The most exhaustive and theoretically grounded survey in this field is (Rahm and Bernstein, 2001), where a classification of the most traditional methods is proposed. Another important classification is (Euzenat and Shvaiko, 2007) where semantic aware matching operators are also considered.

This classification can be organized differentiating the approaches on the basis of some matching criteria.

The combination of these criteria allows to describe all the possible approaches for matching function $\phi$. We can divide the criteria used to define the matching function in two big groups, according to the information item that is considered:

- **Schema level**: In this case the matching is derived on the basis of the metadata describing the elements of the representations analyzed. It can consider only the element itself or an element according to its position in a structure. Often this criteria lacks on providing semantic information because the meta-data itself could not provide enough information to the system to associate a semantics to the element. For example consider dateA and dateB in a table Product representing respectively shipping date and expiring date of a product. It is intuitive to understand that the information provided only by meta-data in not sufficient.

- **Instance level**: In this case the matching is performed by comparing the data contained in the elements. Analysing the data, we can obtain more information associated to the elements considered that can be used to provide semantics to the elements. Following the example of Shipping Date and Expiring Date analysing the instances we discover that dateA is always before dateB and, if we modeled this constrain in the ontology, is easy to associate the right element from the local source $L$ to the element of the global source $G$.

An additional categorization of the approaches can be defined, according to the method used to compare the elements:

- **Element based**: Only a single element is taken into account.

- **Structural based**: An element is analyzed in relation to the elements having a given structural closure to it (i.e. Tables, Data Sources, Foreign Keys, elements in the same table, etc.).

- **Language based**: Many techniques are based on a linguistic approach, evaluating the linguistic closure among names and textual descriptions of elements.

- **Constraint based**: Other techniques evaluate the closure among elements on the basis of constraints such as keys and relationships.

The matching process developed in our system support several matching approaches providing a reach pallet of semantically different operators. The results of the different operators finally needs to be combined somehow by a merging function. At the moment we implemented an early version of the $M_i$ generator which is a combination of instance-level, based
on values-based operator, and schema level approach, based on linguistic-analysys merged by an OWA aggregator (Yager, 1988).

3.1 Identifier Constraints and Attribute Relations

After the generation of $M_t$, to complete the mapping it is necessary to generate also the two sets $M_p$ and $M_o$. As defined in section 2, $M_r$ is the mapping occurring between elements in the local schema $L$; these relations can be derived from a set of Identifier Constraint (IC): a group of matching relations from $M_r$, related each other by means of Boolean operators $\lor$ and $\land$. For example we consider the ontology in fig. 1 as global schema $G$ and the three data sources represented in fig. 2 as local schema $L$. Let’s assume that the concepts in the global schema contain the following attributes:

Customer = \{ FullName, Address, Contact, FiscalCode \}

Employee = \{ FullName, Address, Contact \}

RepairOperation = \{ Date, StartingTime, EndingTime, Type, CustomerSatisfaction \}

Ah-hoc wrappers are used to extract all the useful information about the elements in the two schemas $G$ and $L$. We consider as elements the columns of the tables in $L$ and the attributes of the concepts in $G$. The information provided by the wrappers are: name, type, belonging table or concept, belonging schema and all the available relations between elements.

The matching function analyses the elements extracted and generates the relations reported in table 1. The $\Delta$, in the example, is a pattern-based operator used to discover the country (international prefix) and the province (district number) of a Customer or an Office.

As considered previously, wrappers provide also information about referential integrity constrains extracted from the local sources $L$. The $\land$ operator is used to relate the elements, because it is important to consider all the relations discovered to generate the IC.

In the example in fig. 2 there are two referential integrities that can be extracted from the data base meta-data:

$REL_DB1 = \{ \begin{array}{c} \text{OfficeCode}_D^{DS1} \\
\land \\
\text{OfficeCode}_E^{DS1} \\
\end{array} \} \text{ (IC1)}$

$REL_DB3 = \{ \begin{array}{c} \text{OperationID}_D^{DS3} \\
\land \\
\text{OperationID}_C^{DS3} \\
\end{array} \} \text{ (IC2)}$

To compete the set of IC we need to consider also the semantic relations relating elements belonging to different local representations. We named semantic relations those relations among tables of different sources that carry the same informative value and this way allow to generate virtual join among tables. We generate these ICs considering the matching relations from $M_r$ that refer to elements of the local schema, connecting the relations using the $\land$ operator.

$SEM_{REL\_1} = \{ \begin{array}{c} \text{CostumerNumber}_D^{DS1} \\
\land \\
\text{CostumerNumber}_D^{DS3} \\
\end{array} \} \text{ (IC3)}$

$SEM_{REL\_2} = \{ \begin{array}{c} \text{Department}_D^{DS2} \\
\land \\
\text{Department}_D^{DS3} \\
\end{array} \} \text{ (IC4)}$

$SEM_{REL\_3} = \{ \begin{array}{c} \text{EmployeeNumber}_E^{DS1} \\
\land \\
\text{EmployeeNumber}_C^{DS3} \\
\end{array} \} \text{ (IC5)}$

$SEM_{REL\_4} = \{ \begin{array}{c} \text{FirstName}_E^{DS1} \\
\land \\
\text{ContactFirstName}_C^{DS3} \\
\end{array} \} \text{ (IC6)}$

$SEM_{REL\_5} = \{ \begin{array}{c} \text{LastName}_E^{DS1} \\
\land \\
\text{ContactLastName}_C^{DS3} \\
\end{array} \} \text{ (IC7)}$

Once the set of IC is generated the next step is to discover the relations between elements of the global schema $G$ and elements of the local schema $L$ we call this relations Attribute Relation (AR). Considering always the elements of the set $M_r$, for each element in the global schema $G$, an AR is a binary relation (= or <) with the elements form $L$ that the matching process
Figure 2: The Data Sources used for our example and the relations between them.

Table 1: List of matching relations between meta-data elements in the example, here are listed also the semantic relations between data sources.
4 Mapping

The goal of a mapping is to relate elements having the same informative value in the different representations analyzed.

According to section 2 a mapping can follow the procedural or the declarative approach. This second approach is more flexible because it allows to act on the mapping information during the query processing phase and it can be used to summarize or compose elements of the schemas \( G \) and \( L \).

A mapping relation is then a relation between entities of type 1:1, 1:n, m:1 or n:m. For instance given a set of elements \( \{ e_1, e_2, \ldots, e_m \} \) belonging to a representation \( s_1, s_2, \ldots, s_g \) in \( L \) and a set of elements \( \{ e_1, e_2, \ldots, e_n \} \) belonging to a representation \( g \), a mapping can relate the elements such as for instance, \( e_k \rightarrow e_1 \psi e_2 \psi \ldots \psi e_m \) in a GaV fashion and \( e_k \rightarrow e_1 e_2 e_3 \ldots e_n \) in a LaV fashion.

Where \( \psi \) represents an arbitrary constrain that can range from set-theoretic operators (\( \& \), \( \lor \), \( \ldots \)) to more complex combination \( e_k \rightarrow \phi(T) \) where \( \phi \) is an arbitrary formula that combines the arguments and \( T \) is a series defined as \( T = \{ e_1, e_2, e_3, e_4, e_5, \ldots, e_n \} \) Referring to section 3.1. AR10 is an example of composition.

Following an all-in-one fashion we exploit the searching space given by modeling our information as an FCA lattice, processing the lattice to generate the set of mapping \( M_p \) and the relations \( M_o \) for the elements involved in the mapping.

4.1 FCA-based Mapping Generation

The mapping generation process is based on Formal Concept Analysis (FCA) (Ganter et al., 2005) to discover relations between objects \( O_1 \) and \( O_2 \) of the two schemas \( L \) and \( G \).

FCA has strong mathematical foundations; it is a branch of lattice theory and its goal is to generate a lattice representing the relations between objects and attributes. The relations between the objects and the attributes are modeled using a formalism called FCA Context.

An FCA context can be represented as a matrix with objects (rows) and attributes (columns), the matrix is used to model relations between objects and attributes: where a relation exists a true Boolean value is inserted in the respective cell, a false value otherwise.

Given an FCA context \( \mathcal{R} \) as described above we can define it as a triple \( \mathcal{R} = < O, A, R > \) where \( O = o_1, o_2, \ldots, o_n \) are the set of objects, \( A = a_1, a_2, \ldots, a_m \) the set of attributes and \( R \) the set of relations \( r(x,y) \) is \( o_x \bowtie a_y \) between elements of \( A \) and \( O \) defined as:

\[
\begin{align*}
\text{true} & \quad \text{if } a_y \text{ is an attribute of } o_x \\
\text{false} & \quad \text{otherwise}
\end{align*}
\]

A formal concept in the concept lattice is defined to be a pair \( (O_i, A_i) \) such that: (i) if \( O_i \subseteq O \); (ii) \( A_i \subseteq A \); (iii) every object in \( O_i \) has every attribute in \( A_i \); (iv) for every object \( o_x \) in \( O \) that is not in \( O_i \), there is an attribute \( a_y \) in \( A_i \) that the object does not have; (v) for every attribute \( a_y \) in \( A_i \) that is not in \( A \), there is an object \( o_x \) in \( O_i \) that does not have that attribute. \( O_i \) is called the extent of the concept, \( A_i \) the intent.

These concepts can be partially ordered by inclusion: if \( (O_i, A_i) \) and \( (O_j, A_j) \) are concepts, we define a partial order \( \preceq \) by saying that \( (O_i, A_i) \preceq (O_j, A_j) \) whenever \( O_i \subseteq O_j \). Equivalently, \( (O_i, A_i) \preceq (O_j, A_j) \) whenever \( A_j \subseteq A_i \).
In this section we provided just an introduction to FCA for the sake of clearness. For a more exhaustive theoretically grounded coverage, please refer to (Ganter et al., 2005).

To generate the FCA lattice we define a set of clusters generated by the ICs and ARs: a cluster \( c_i \) is defined by all the elements that belong to a relation IC or AR. For example in case of the relation object referring Employee/Customer the correspondent cluster \( c_i \) is defined as:

\[
\begin{align*}
\text{Employee/Customer} & = \{ \text{CustomerName} \} \\
\text{Employee/Customer} & = \{ \text{ContactFirstName} \} \\
\text{Employee/Customer} & = \{ \text{ContactLastName} \} \\
\text{Employee/Customer} & = \{ \text{Employee} \} \\
\text{Employee/Customer} & = \{ \text{Customer} \} \\
\text{Employee/Customer} & = \{ \text{Employee} \} \\
\text{Employee/Customer} & = \{ \text{Customer} \} \\
\text{Employee/Customer} & = \{ \text{Employee} \} \\
\end{align*}
\]

In Table 2 we list the relations (AR) relating elements from the local schemas \( L \) to elements of the global schema \( G \).

Applying lattice theories on \( \mathcal{R} \) a lattice \( \mathcal{Z} \) is generated. Figure 3 shows the lattice representation of the formal context represented in Table 3 generated using Concept Explorer (http://conexp.sourceforge.net/).

![Figure 3: FCA Concept Lattice](Image 337x198 to 493x314)

We assume that the attributes \( A \) are the clusters \( c_1, c_2, \ldots, c_n \) of the set \( C \) and the objects \( O \) are the union of the set of object \( o_s \) from \( s_1 \) in \( L \) (tables in a data source) and the set of objects \( o_g \) from \( G \) (concepts of the ontology).

The mapping process then generates a formal context \( \mathcal{R} = < O, C, R > \) where \( R \), the set of relations \( r_{(x, y)} = o_i \rhd c_j \) is defined as:

\[
\begin{align*}
o_i \rhd c_j & = \begin{cases} 
true & \text{if the cluster } c_j \text{ contains an attribute } a_k \text{ of } o_i \\
false & \text{otherwise}
\end{cases}
\end{align*}
\]

Figure 3: FCA Concept Lattice

The lattice generated is processed in order to discover semantic relations between objects \( o_s \) from the source schema \( s_1 \) in \( L \) (the tables of a data source) and the set of objects \( o_g \) from \( G \) (the concepts of the ontology).
The mapping algorithm analyses the FCA concept lattice and generates, for each objects $o_g$ of $G$, a set of element $T_{o_g} = \{t_{e_1}, t_{e_2}, ..., t_{e_j}\}$ where $t_{e_j}$ is related to an attribute of a concept of $G$ and is defined as:

$$t_{e_j} = <c_k, W>$$

with:

$$c_k \in C \forall e_g \in c_k \exists W = \bigcup_{j=0}^{m} e_{d_j} \forall e_{d_j} \in c_k$$

Once we generated the clusters, applying the lattice generation formula to the clusters, our algorithm generates the FCA lattice showed in fig. 5. This lattice reports all the information extracted during the matching process and these information are distributed in a highly structured searching space that will be the input of our mapping generation algorithm. The lattice is covered by the mapping generation algorithm in an iterative way, and the use of a FCA as formalism to represent the information is a sound and strongly mathematical-based representation that allow to flow easily between the relations that occur between concepts and elements. For example, considering the concept $Customer_{ONTO}$, the first step is to get the intent of the object to map. Selecting $Customer_{ONTO}$ in the FCA lattice we obtain the set of intents $Address$, $Fullname$, $FiscalCode$ and $Contact$ that refers to the respective clusters. Now, for each intent considered, we extract the extent (not considering the elements of $G$) that represents the target objects of the selected intent. Refering to the example in fig. 4 the list of extents to consider for the intent $Fullname$ will be:

$$T(Fullname_{Customer_{ONTO}}) = \{Customer_{DS1}, Employee_{DS1}\}$$

Following the same procedure we obtain the remaining $T_i$:

$$T(Contact_{Customer_{ONTO}}) = \{Customer_{DS1}, Employee_{DS1}, Office_{DS1}\}$$

$$T(Address_{Customer_{ONTO}}) = \{Customer_{DS1}, Office_{DS1}\}$$

Table 3: The FCA context table

<table>
<thead>
<tr>
<th></th>
<th>hunts</th>
<th>fly</th>
<th>bird</th>
<th>mammal</th>
<th>swim</th>
<th>fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lion</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finch</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eagle</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hare</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ostrich</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bat</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shark</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Penguin</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Whale</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Selecting the object $Customer_{ONTO}$ the intents $Address$, $FiscalCode$, $Fullname$ and $Contact$ are obtained (above), for each intent the target objects are discovered. In case of $Fullname$ the extents are $Customer_{ONTO}$, $Customer_{DS1}$, $Employee_{ONTO}$ and $Employee_{DS1}$ (below)

$$T(FiscalCode_{Customer_{ONTO}}) = \{Customer_{DS1}\}$$

The set $T_{o_g}$ of elements $t_{e_j}$ needs to be semantically analysed and pruned off from the redundant information.

The set $T_{o_g}$ is pruned by applying a process that removes from the set $T_{o_g}$ the elements that do not share any equal instance. The algorithms performs a set of queries and analyse the results to decide which elements $e_{s_n}$ of $s_n$ in $L$ are not semantically related to the element $e_{s_g}$ of $G$. We will avoid the details of the queries preformed to disambiguate the elements be-
cause the semantic queries are out of the scope of this paper. After the pruning process the set \( T_{e} \) will be:

\[
\begin{align*}
T_{Customer_{ONTO}} &= \{ \text{Customer}_{DS1} \} \\
T_{Employee_{ONTO}} &= \{ \text{Employee}_{DS1} \} \\
T_{Address_{ONTO}} &= \{ \text{Office}_{DS1} \} \\
T_{Task_{ONTO}} &= \{ \text{Department}_{DS2} \}
\end{align*}
\]

for the concepts Customer and Employee respectively. Once the set of \( T_{e} \) is pruned from the redundant information it is possible to convert the set in the mappings \( M_{p} \) and \( M_{o} \). The conversion process is performed substituting the objects in \( T_{e} \) (the tables of the local schemas) with its correspondent in the set of ARs. Referring to the previous example we obtain the relations in table 4.

Which is the set of \( M_{p} \) for the concepts Employee and Customer. To complete the mapping we need to generate the set \( M_{o} \) that is produced according to the tables involved in the mapping \( M_{p} \). Considering the set of ICs generated previously the set of \( M_{o} \) is empty in case of the concept Customer, because all the attributes are mapped on a single table, in case of the concept Employee the tables involved are: Employee_{DS1}, Office_{DS1} and Department_{DS2} and then the set \( M_{o} \) in this case is composed by all the ICs that refers to the tables considered in the mapping. It is important to underline that all the tables of the IC need to be present in the mapping. The set \( M_{o} \) in case of Employee is: \( M_{o} = \{ [IC1], [IC4] \} \).

The resulting sets are the mapping \( M_{p} \) and \( M_{o} \) that concludes the mapping generation process. The discovered mapping \( M \) needs to be validated: if the mapping does not return any result from the query engine or the query can not be resolved then the mapping is not considered to be correct, the wrong mapping is passed in the mapping generation process and an alternative mapping is generated.

### 5 Conclusions

This paper addressed the issue of managing a variety of matching operator in a complex Data Integration system executing a semiautomatic process. A solution based on a categorization of matching operators that allow to group similar attributes on a semantic rich form. This way we define all the information need in order to create a mapping. Then Mapping Generation is activated only on those set of elements that can be queried without violating any integrity constraints on data.

The results of the test will be reported in a separate paper. Several public data sources and the correspondent ontological representation can be found online and they can be used for an evaluation by comparing the mappings generated by our tool with mapping generated by a domain expert. This way we can compare our tool with others well known data integration systems (COMA++, OntoBuilder, Harmony, Mafra) by exploiting classical Information Retrieval quality measures such as Precision and Recall. Moreover, the matching process is a key factor for the quality of the final mapping, then we performed an additional evaluation based on the benchmark test of the Ontology Alignment Evaluation Initiative contest 2007 (OAEI, http://oaei.ontologymatching.org/). The use of an OWA aggregator in the matching process revealed the limits of this approach during this last test. An OWA aggregator is not decisional and furthermore we need a method that is capable to consider also the semantic of different matching operators. Future work will focus on the use of logic based decisional process that will build a knowledge base starting from the results of the matching operators and will return the final matching as result of reasoning process over the knowledge base.

### Acknowledgements

This work was partly funded by the Italian Ministry of Research under FIRB contract n. RBNE05FKZ2_004, TEKNE. This work is also partially supported by British Telecom (BT) research and venturing.
Table 4: Mapping relations for the concepts Customer and Employee generated by the FCA-mapping generator

REFERENCES


