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* approch the mapping is provided on G objects by using a L vocabulary while in the *Local as View* approch 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 information need in order to create a mapping. Also Mapping Generation is activated only on those set of elements that can be queried without violating any integrity constraints. Our system is named *Ontology Driven Data Integration* (ODDI), it is based on *Formal Concept Analysis* (FCA)(Ganter et al., 2005) as searching space in order to discover concept-level relations for Mapping Generation, and uses an ontology as data access layer. Using an ontology as common conceptualization brings several benefits but the more relevant is that due to the sound logic basis it is possible to perform reasoning task on the knowledge base such as *Consistency Checking* and *Classification* (Cui et al., 2007).

The paper is structuerd as follows: Section 2 introduces the formally a generic Data Integration System, focusing then to our definition of mapping; then in Section 3 the matching process is described, providing initially our formalization and then a categorization of the traditional matching operators. Section 4 describes the mapping generation module, focusing on the use of FCA as a formalism for representing the information. The paper is enriched with an example of the generation of the FCA lattice starting from a local schema S and a global schema G. Section 5 concludes the paper, outlining conclusions.

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_t, 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_a 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_t, given G and L. The second module generates M_p and M_o, by representing M_t as an FCA used as searching space to find semantic relations between elements of G, and L.

3 Matching

Matching (M_t) 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:

$$e^i_{s_k} \cong^{\delta} e^j_g$$

Where $\cong$ is a binary relation from the set of the following relations: equality, inclusion and specification ($=$, $<$, $\subset$). While δ, 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 strenght of the relation. Then elements of m_t are: $e^i_{s_k} = e^j_g$ if the data items carried by $e^i_{s_k}$ satisfy e^j_g with a sound and complete information (i.e. *name* and*firstName*). Or in case of inclusion $e^i_{s_k} < e^i_g$ if the data items carried by $e^i_{s_k}$ provide a portion of the information required for satisfying e^j_g (i.e. *street* and *address*). $e^i_g < e^i_{s_k}$ if the data items carried by $e^i_{s_k}$ contain more information than the one required for satisfying e^j_g (i.e. *street* and *address*). Or, in case of specification, $e^i_{s_k} \subset e^i_g$ if the data items carried by $e^i_{s_k}$ satisfy e^j_g 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 Δ as a symbol for meaning these transformation. So we can have a matching such as:

$$\Delta e^i_{s_k} \cong^{\delta} e^i_g$$

if the information carried by $e^i_{s_k}$ allow a transformation that satisfy the semantics of e^i_g. As an example, if e^i_g represent the nationality of a person and $e^j_{s_h}$ 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 δ are the output of a matching function $\phi(e^i_{s_k}, e^j_{s_h})$ that evaluates the semantic relations between two elements $e^i_{s_k}$ and $e^j_{s_h}$.

A variety of methods for implementing the matching function ϕ were proposed in the literature in order to define correspondences among elements belonging to different representations.

The most exhaustive and theorically grounded survey in this field is (Rahm and Bernstein, 2001), where a classification of the most traditional methods is provided. 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 ϕ. 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 enought 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_t generator which is a combination of instance-level, based on values-based operator, and schema level approach, based on linguistic-analisys 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_o 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_t, related each other by means of Boolean operators $\vee$ and $\wedge$. 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:

$$Customer = \left\{ \begin{array}{l} FullName, \\ Address, \\ Contact, \\ FiscalCode \end{array} \right\}.$$

$$Employee = \left\{ \; FullName, Address, Contact \; \right\}.$$

$$RepairOperation = \left\{ \begin{array}{l} Date, \\ StartingTime, \\ EndingTime, \\ Type, \\ CustomerSatisfaction \end{array} \right\}.$$

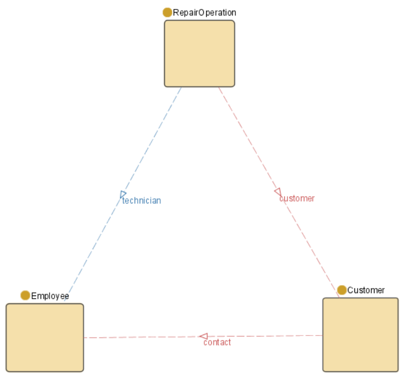


Figure 1: A simple ontology used for our example

Ah-hoc wrappers are used 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 Δ, in the example, is a pattern-based operator used to discover the country (international prefix) and the province (district number) of a *Costumer* or an *Office*.

As considered previously, wrappers provide also information about referential integrity constrains extracted from the local sources L. The $\wedge$ 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:

$$REL\_DB1 = \left\{ \begin{array}{l} OfficeCode_{DS1}^{Office} \\ \wedge \\ OfficeCode_{DS1}^{Employee} \end{array} \right\} (IC1)$$

$$REL\_DB3 = \left\{ \begin{array}{l} OperationID_{DS3}^{RepairOperation} \\ \wedge \\ OperationID_{DS3}^{CostumerSatisfaction} \end{array} \right\} (IC2).$$

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_t that refer to elements of the local schema, connecting the relations using the $\wedge$ operator.

$$SEM\_REL\_1 = \left\{ \begin{array}{l} CostumerNumber_{DS1}^{Costumers} \\ \wedge \\ CostumerID_{DS3}^{CostumerSatisfaction} \end{array} \right\} (IC3)$$

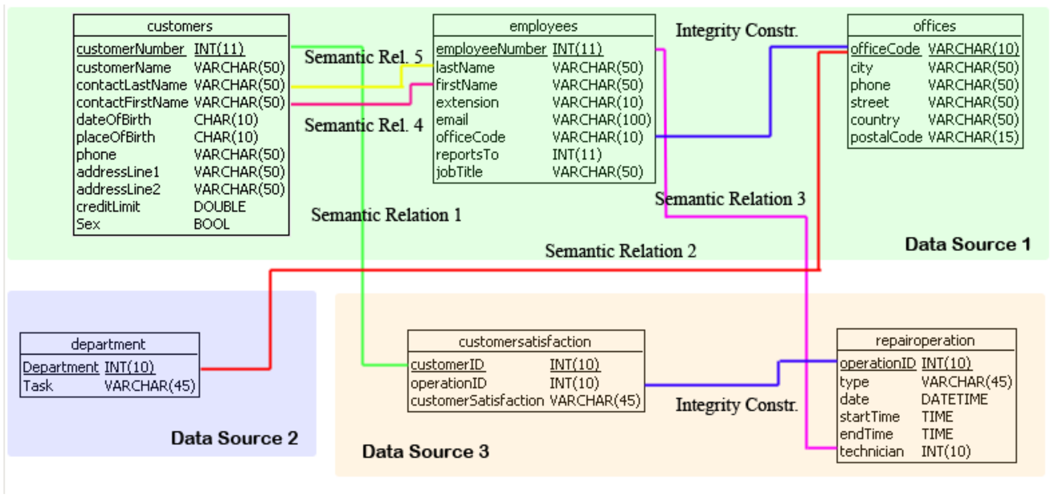
$$SEM\_REL\_2 = \left\{ \begin{array}{l} Department_{DS2}^{Department} \\ \wedge \\ OfficeCode_{DS1}^{Office} \end{array} \right\} (IC4)$$

$$SEM\_REL\_3 = \left\{ \begin{array}{l} EmployeeNumber_{DS1}^{Employee} \\ \wedge \\ Technician_{DS3}^{RepairOperation} \end{array} \right\} (IC5)$$

$$SEM\_REL\_4 = \left\{ \begin{array}{l} FirstName_{DS1}^{Employee} \\ \wedge \\ ContactFirstName_{DS1}^{Costumer} \end{array} \right\} (IC6)$$

$$SEM\_REL\_5 = \left\{ \begin{array}{l} LastName_{DS1}^{Employee} \\ \wedge \\ ContactLastName_{DS1}^{Costumer} \end{array} \right\} (IC7).$$

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_t, for each element in the global schema G, an AR is a binary relation ($=$ or $<$) with the elements form L that the matching process



customers	
customerNumber	INT(11)
customerName	VARCHAR(50)
contactLastName	VARCHAR(50)
contactFirstName	VARCHAR(50)
dateOfBirth	CHAR(10)
placeOfBirth	CHAR(10)
phone	VARCHAR(50)
addressLine1	VARCHAR(50)
addressLine2	VARCHAR(50)
creditLimit	DOUBLE
Sex	BOOL

employees	
employeeNumber	INT(11)
lastName	VARCHAR(50)
firstName	VARCHAR(50)
extension	VARCHAR(10)
email	VARCHAR(100)
officeCode	VARCHAR(10)
reportsTo	INT(11)
jobTitle	VARCHAR(50)

offices	
officeCode	VARCHAR(10)
city	VARCHAR(50)
phone	VARCHAR(50)
street	VARCHAR(50)
country	VARCHAR(50)
postalCode	VARCHAR(15)

department	
Department	INT(10)
Task	VARCHAR(45)

customersatisfaction	
customerID	INT(10)
operationID	INT(10)
customerSatisfaction	VARCHAR(45)

repairoperation	
operationID	INT(10)
type	VARCHAR(45)
date	DATETIME
startTime	TIME
endTime	TIME
technician	INT(10)

Figure 2: The Data Sources used for our example and the relations between them

$FirstName^{Employee}_{DS1}$	$=$	$ContactFirstName^{Customer}_{DS1}$
$LastName^{Employee}_{DS1}$	$=$	$ContactLastName^{Customer}_{DS1}$
$OfficeCode^{Office}_{DS1}$	$=$	$Department^{Department}_{DS2}$
$EmployeeNumber^{Employee}_{DS1}$	$=$	$Technician^{RepairOperation}_{DS3}$
$CustomerNumber^{Customer}_{DS1}$	$=$	$CustomerID^{CustomerSatisfaction}_{DS3}$
$FullName^{Employee/Customer}_{Onto}$	$=$	$CustomerName^{Customer}_{DB1}$
$Phone^{Customer}_{DS1}$	$=$	$TelePhone^{Office}_{DS1}$
$Task^{Employee}_{Onto}$	$=$	$Task^{Department}_{DB2}$
$Date^{RepairOperation}_{Onto}$	$=$	$Date^{RepairOperation}_{DB3}$
$Start^{RepairOperation}_{Onto}$	$=$	$Start^{RepairOperation}_{DB3}$
$End^{RepairOperation}_{Onto}$	$=$	$End^{RepairOperation}_{DB3}$
$Type^{RepairOperation}_{Onto}$	$=$	$Type^{RepairOperation}_{DB3}$
$SatisfactionLevel^{RepairOperation}_{Onto}$	$=$	$CustomerSatisfaction^{CustomerSatisfaction}_{DB3}$
$FirstName^{Employee}_{DS1}$	$<$	$CustomerName^{Customer}_{DS1}$
$LastName^{Employee}_{DS1}$	$<$	$CustomerName^{Customer}_{DS1}$
$FirstName^{Employee}_{DS1}$	$<$	$FullName^{Employee/Customer}_{Onto}$
$LastName^{Employee}_{DS1}$	$<$	$FullName^{Employee/Customer}_{Onto}$
$AddressLine1^{Customer}_{DS1}$	$<$	$Address^{Employee/Customer}_{Onto}$
$AddressLine2^{Customer}_{DS1}$	$<$	$Address^{Employee/Customer}_{Onto}$
$Street^{Office}_{DS1}$	$<$	$Address^{Employee/Customer}_{Onto}$
$City^{Office}_{DS1}$	$<$	$Address^{Employee/Customer}_{Onto}$
$PostalCode^{Office}_{DS1}$	$<$	$Address^{Employee/Customer}_{Onto}$
$Phone^{Customer}_{DS1}$	$\subset$	$Contact^{Employee/Customer}_{Onto}$
$Email^{Employee}_{DS1}$	$\subset$	$Contact^{Employee/Customer}_{Onto}$
$TelePhone^{Office}_{DS1}$	$\subset$	$Contact^{Employee/Customer}_{Onto}$
$\Delta(TelePhone^{Office}_{DS1})$	$<$	$Address^{Employee/Customer}_{Onto}$
$\Delta(Phone^{Customer}_{DS1})$	$<$	$Address^{Employee/Customer}_{Onto}$

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

considers similar. If more than one element from L can be associated to the element in G we relate by the $\wedge$ operator the elements from L belonging to the same table or belonging to tables related by an IC. Then we relate these groups using the $\vee$ operator. In case of relations of type $\subset$ the elements in AR are related using the only $\vee$ operator. Referring to the example the relations generated are reported in table 2.

Referring to table 1 is possible to see that the element $FiscalCode_{ONTO}^{Customer}$ is not present and then a relation can not be generated, so we can presume that this information is missing in the local schema. Anyway it is possible to overcome this limit assuming that the value of this element is defined as the output of a function Fc with *fullname*, *birthplace*, *birthdate* and *sex* as arguments. This function can not be defined automatically by the system for this reason it is defined externally (for example as a Java class) and the system refers to it by an annotation on the element $FiscalCode_{ONTO}^{Customer}$. This way we can generate the AR for the missing attribute. Analysing the annotation we extract the names of the arguments and by applying the matching function with the elements of the local schema L we discover new matching relations that can be used as arguments of the function Fc. With the new matching relations it is possible to define the AR related to $FiscalCode$:

$$FiscalCode_{ONTO}^{Customer} = Fc\left(\begin{array}{l} CustomerName_{DS1}^{Customer} \\ DateOfBirth_{DS1}^{Customer} \\ PlaceOfBirth_{DS1}^{Customer} \\ Sex_{DS1}^{Customer} \end{array}\right) \quad (AR10)$$

The ICs and the ARs are then used as fundamental information to generate the Formal Concept Analysis lattice used to generate the remaining mappings M_o and M_p.

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 $\{e_{s_a}^1, e_{s_a}^2 ... e_{s_m}^n\}$ belonging to a representation $s_a, ..., s_n$ in L and a set of elements $\{e_g^1, e_g^2, ... e_g^n\}$ belonging to a representation G, a mapping can relate the elements such as for instance, $e_g^k \rightarrow e_{s_a}^1 \psi e_{s_b}^1 \psi ... \psi e_{s_m}^n$ in a GaV fashion and $e_{s_j}^i \rightarrow e_g^1 \psi e_g^2 \psi ... \psi e_g^n\}$ in a LaV fashion.

Where ψ represents an arbitrary constrain that can range from set-theoretic operators $(\cap, \cup, ...)$ to more complex combination $e_g^k \rightarrow \phi(T)$ where ϕ is an arbitrary formula that combines the arguments and T is a series defined as $T = \{e_{s_a}^1, e_{s_a}^2 \phi e_{s_c}^5, e_{s_b}^1 \phi e_{s_b}^2, ..., e_{s_m}^n\}$ 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 ($O_{s_n}^i$ and O_g^j) 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 $\Re$ as described above we can define it as a triple $\Re = < O, A, R >$ where $O = o_1, o_2, ..., o_n$ are the set of objects, $A = a_1, a_2, ..., a_m$ the set of attributes and R the set of relations $r_{(x,y)} = o_x \bowtie a_y$ between elements of A and O defined as:

$$o_x \bowtie a_y = \begin{cases} true & \text{if } a_y \text{ is an attribute of } o_x \\ false & \text{otherwise} \end{cases}$$

A *formal concept* in the concept lattice is defined to be a pair (O_i, A_i) such that: (i) $O_i \subseteq O$; (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_x in A that is not in A_i, there is an object o_y 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 $\leq$ by saying that $(O_i, A_i) \leq (O_j, A_j)$ whenever $O_i \subseteq O_j$. Equivalently, $(O_i, A_i) \leq (O_j, A_j)$ whenever $A_j \subseteq A_i$.

$$
\begin{aligned}
Task_{ONTO}^{Employee} &= \left\{ Task_{DS2}^{Department} \right\} && (AR1)\\
Type_{ONTO}^{RepairOperation} &= \left\{ Type_{DS3}^{RepairOperation} \right\} && (AR2)\\
CustomerSatisfaction_{ONTO}^{RepairOperation} &= \left\{ SatisfactionLevel_{DS3}^{CostumerSatisfaction} \right\} && (AR3)\\
Date_{ONTO}^{RepairOperation} &= \left\{ Date_{DS3}^{RepairOperation} \right\} && (AR4)\\
StartingTime_{ONTO}^{RepairOperation} &= \left\{ StartingTime_{DS3}^{RepairOperation} \right\} && (AR5)\\
EndingTime_{ONTO}^{RepairOperation} &= \left\{ StartingTime_{DS3}^{RepairOperation} \right\} && (AR6)\\
Fullname_{ONTO}^{Employee/Customer} &= \left\{ \begin{array}{l} CustomerName_{DS1}^{Customer} \\ \vee \\ (FirstName_{DS1}^{Employee} \wedge LastName_{DS1}^{Employee}) \\ \vee \\ (ContactLastName_{DS1}^{Costumer} \wedge ContactFirstName_{DS1}^{Costumer}) \end{array} \right\} && (AR7)\\
Contact_{ONTO}^{Employee/Customer} &= \left\{ \begin{array}{l} TelePhone_{DS1}^{Office} \\ \vee \\ Email_{DS1}^{Employee} \\ \vee \\ Phone_{DS1}^{Customer} \end{array} \right\} && (AR8)\\
Address_{ONTO}^{Employee/Customer} &= \left\{ \begin{array}{l} (AddressLine1_{DS1}^{Customer} \wedge AddressLine2_{DS1}^{Customer} \wedge \\ \Delta(Phone_{DS1}^{Customer})) \\ \vee \\ (Street_{DS1}^{Office} \wedge City_{DS1}^{Office} \wedge PostalCode_{DS1}^{Office} \wedge \\ \Delta(TelePhone_{DS1}^{Office})) \end{array} \right\} && (AR9).
\end{aligned}
$$

Table 2: List of Attribute Relations (AR) relating elements from the local schemas L to elements of the global schema G.

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*i* or AR*i*. For example in case of the relation object referring $Fullname_{ONTO}^{Employee/Customer}$ the correspondent cluster c_i is defined by:

$$
c_{FULLNAME} \left\{ \begin{array}{l} FullName_{ONTO}^{Employee} \\ FullName_{ONTO}^{Customer} \\ CustomerName_{DS1}^{Customer} \\ FirstName_{DS1}^{Employee} \\ LastName_{DS1}^{Employee} \\ ContactLastName_{DS1}^{Costumer} \\ ContactFirstName_{DS1}^{Costumer} \end{array} \right\}
$$

We assume that the attributes A are the clusters $c_1, c_2, ..., c_n$ of the set C and the objects O are the union of the set of object o_{s_n} from s_n 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 $\mathfrak{R} =< O, C, R >$ where R, the set of relations $r_{(x,y)} = o_x \bowtie c_y$is defined as:

$$
o_i \bowtie c_j = \begin{cases} true & \text{if the cluster } c_j \text{ contains} \\ & \text{an attribute } a_k \text{ of } o_i \\ false & \text{otherwise} \end{cases}
$$

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

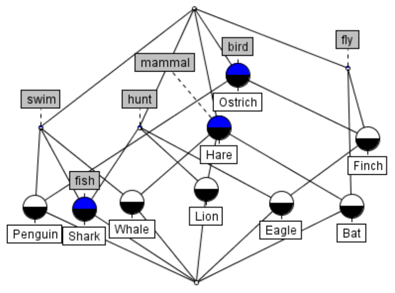


Figure 3: FCA Concept Lattice

The lattice generated is processed in order to discover semantic relations between objects o_s^n from the source schema s_n in L (the tables of a data source) and the set of objects o_g from G (the concepts of the ontology).

	hunts	fly	bird	mammal	swim	fish
Lion	×			×		
Finch		×	×			
Eagle	×	×	×			
Hare				×		
Ostrich			×			
Bat		×		×		
Shark	×				×	×
Penguin			×		×	
Whale				×	×	

Table 3: The FCA context table

The mapping algorithm analyses the FCA concept lattice and generates, for each objects o_g of G, a set of element $T_{o_g} = \{t_{e_g^1}, t_{e_g^2}, ..., t_{e_g^j}\}$ where $t_{e_g^j}$ is related to an attribute of a concept of G and is defined as:

$$t_{e_g^j} = < c_k, W >$$

with:

$$c_k \in C | \forall e_g^x \in c_k \exists W = \bigcup_{j=0..n}^{i=0..m} e_{s_j}^i \forall e_{s_j}^i \in c_k$$

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 out 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. Referring to the example in fig. 4 the list of extents to consider for the intent *FullName* will be:

$$T(FullName_{ONTO}^{Customer}) = \{Customer_{DS1}, Employee_{DS1}\}$$

Following the same procedure we obtain the remaining T_i:

$$T(Contact_{ONTO}^{Customer}) = \left\{ \begin{array}{c} Customer_{DS1}, \\ Employee_{DS1}, \\ Office_{DS1} \end{array} \right\}$$

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

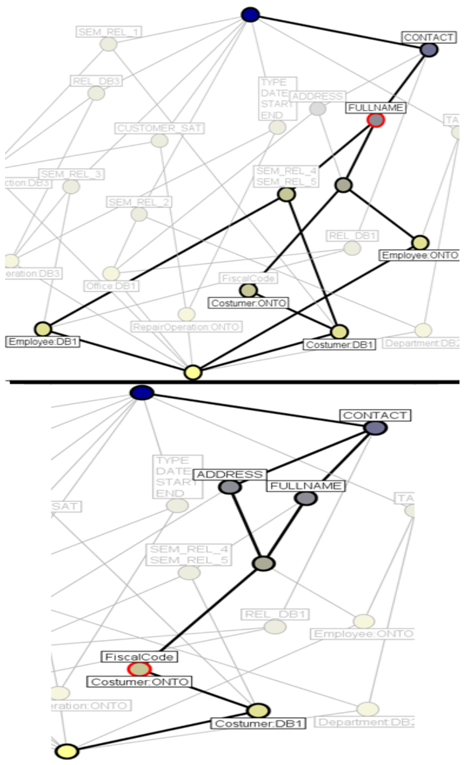


Figure 4: Selecting the object *Customer:ONTO* the intents *Address*, *FiscalCode*, *FullName* and *Contact* are obtained (above), for each intent the target objects are discovered. In case of *Fullname* the extents are *Customer:ONTO*, *Customer:DB1*, *Employee:ONTO* and *Employee:DB1* (below)

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

The set T_{o_g} of elements $T_{e_g^i}$ 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_g of G. We will avoid the details of the queries preformed to disambiguate the elements be-

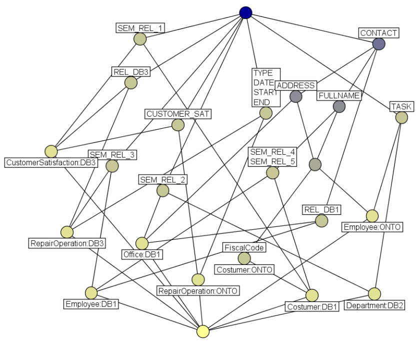


Figure 5: The FCA lattice generated from the schemas of the example

cause the semantic queries are out of the scope of this paper. After the pruning process the set T_{o_g} will be:

$$T_{Customer_{ONTO}} \left\{ \begin{array}{l} T(FullName^{Customer}_{ONTO}) = \{Customer_{DS1}\} \\ T(Contact^{Customer}_{ONTO}) = \{Customer_{DS1}\} \\ T(Address^{Customer}_{ONTO}) = \{Customer_{DS1}\} \\ T(FiscalCode^{Customer}_{ONTO}) = \{Customer_{DS1}\} \end{array} \right\}$$

$$T_{Employee_{ONTO}} \left\{ \begin{array}{l} T(FullName^{Employee}_{ONTO}) = \{Employee_{DS1}\} \\ T(Contact^{Employee}_{ONTO}) = \left\{ \begin{array}{l} Employee_{DS1} \\ Office_{DS1} \end{array} \right\} \\ T(Address^{Employee}_{ONTO}) = \{OfficeDS1\} \\ T(Task^{Employee}_{ONTO}) = \{DepartmentDS2\} \end{array} \right\}$$

for the concepts *Customer* and *Employee* respectively. Once the set of T_{o_g} 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_{o_g} (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.

$$T_{Customer_{ONTO}} \left\{ \begin{array}{l} T(FullName^{Customer}_{ONTO}) = CustomerName^{Customer}_{DS1} \\ T(Contact^{Customer}_{ONTO}) = Phone^{Customer}_{DS1} \\ T(Address^{Customer}_{ONTO}) = \left(\begin{array}{l} AddressLine1^{Customer}_{DS1} \\ \wedge \\ AddressLine2^{Customer}_{DS1} \\ \wedge \\ \Delta(Phone^{Customer}_{DS1}) \end{array} \right) \\ T(FiscalCode^{Customer}_{ONTO}) = Fc \left(\begin{array}{l} CustomerName^{Customer}_{DS1}, \\ DateOfBirth^{Customer}_{DS1}, \\ PlaceOfBirth^{Customer}_{DS1}, \\ Sex^{Customer}_{DS1} \end{array} \right) \end{array} \right\}$$

$$T_{EmployeeONTO} \left\{ \begin{array}{l} T(FullName^{Employee}_{ONTO}) = (FirstName^{Employee}_{DS1} \wedge LastName^{Employee}_{DS1}) \\ T(Contact^{Employee}_{ONTO}) = (TelePhone^{Office}_{DS1} \vee Email^{Employee}_{DS1}) \\ T(Address^{Employee}_{ONTO}) = \left(\begin{array}{l} Street^{Office}_{DS1} \\ \wedge \\ City^{Office}_{DS1} \\ \wedge \\ PostalCode^{Office}_{DS1} \\ \wedge \\ \Delta(TelePhone^{Office}_{DS1}) \end{array} \right) \\ T(Task^{Employee}_{ONTO}) = Task^{Department}_{DS2} \end{array} \right\}$$

Table 4: Mapping relations for the concepts *Customer* and *Employee* generated by the FCA-mapping generator

REFERENCES

Abiteboul, S. and Duschka, O. M. (1998). Complexity of answering queries using materialized views. pages 254–263.

Braun, P., Lötzbeyer, H., Schätz, B., and Slotosch, O. (2000). Consistent integration of formal methods. In *TACAS '00: Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 48–62, London, UK. Springer-Verlag.

Calvanese, D., Lenzerini, M., and Nardi, D. (1998). Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263.

Cui, Z., Damiani, E., and Leida, M. (2007). Benefits of ontologies in real time data access. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES , vol., no., pp.392-397, 21-23 Feb. 2007.*

Duschka, O. M., Genesereth, M. R., and Levy, A. Y. (2000). Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73.

Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*. Springer-Verlag, Heidelberg (DE).

Ganter, B., Stumme, G., and Wille, R., editors (2005). *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*. Springer.

Grahne, G. and Mendelzon, A. O. (1999). Tableau techniques for querying information sources through global schemas. *Lecture Notes in Computer Science*, 1540:332–347.

Hakimpour, F. and Geppert, A. (2002). Global schema generation using formal ontologies.

Halevy, A. Y. (2001). Answering queries using views: A survey. *VLDB Journal: Very Large Data Bases*, 10(4):270–294.

Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., and Zhuge, Y. (1995). The stanford data warehousing project. *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing*, 18(2):41–48.

Lenzerini, M. (2002). Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA. ACM Press.

Parent, C. and Spaccapietra, S. (1998). Issues and approaches of database integration. *Commun. ACM*, 41(5es):166–178.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190.