A Semantic Approach to ETL Technologies

Sonia Bergamaschi\textsuperscript{a}, Francesco Guerra\textsuperscript{a}, Mirko Orsini\textsuperscript{a}, Claudio Sartori\textsuperscript{b}, Maurizio Vincini\textsuperscript{a}

\textsuperscript{a}Università di Modena e Reggio Emilia, Via Università 4, 41121 Modena - Italy
\textsuperscript{b}Università di Bologna Viale Risorgimento 2, 40136 Bologna - Italy

Abstract

Data warehouse architectures rely on extraction, transformation and loading (ETL) processes for the creation of an updated, consistent and materialized view of a set of data sources. In this paper, we aim to support these processes by proposing a tool for the semi-automatic definition of inter-attribute semantic mappings and transformation functions. The tool is based on semantic analysis of the schemas for the mapping definitions amongst the data sources and the data warehouse, and on a set of clustering techniques for defining transformation functions homogenizing data coming from multiple sources. Our proposal couples and extends the functionalities of two previously developed systems: the MOMIS integration system and the \textit{RELEVANT} data analysis system.

Keywords: Semantic ETL, semantic mapping, cluster analysis

1. Introduction

The term “Enterprise Information System” (EIS) refers to the technological framework enabling organizations to integrate, coordinate and analyze their business processes. The data warehouse is the EIS component supporting business intelligence analysis processes on data related to all the enterprise activities. Since there are multiple possible data sources in an enterprise, \textit{extraction, transformation and loading (ETL)} processes for building and populating the data warehouse
are needed. The ETL supports (a) the identification of relevant information at the source side, (b) the extraction of this information, (c) the customization and integration of the information coming from multiple sources into a common format, (d) the cleaning of the resulting data set on the basis of database and business rules, and (e) the propagation of the data to the data warehouse and/or data marts [18].

To know if a particular information is stored in the enterprise data sources and where this information is stored is crucial for providing an accurate analysis of the enterprise processes. Despite the importance of this knowledge, enterprises typically do not codify in any formal way this information. This implies that in general the ETL processes are manually executed and as more accurate as they are led by designers that know the sources and can take into account constraints and requirements which are usually expressed in natural language in the form of comments and documentations.

In the context of traditional databases, this fact does not represent a real big issue: 1) the processes requiring the manual user intervention involve data source schemata that are generally fixed. Thus, designers are asked to set up the ETL processes once per data source (during the start up phase); 2) all the data sources collected in the data warehouses typically belong to the same company and are known and managed by the same designers. Thus, technical documentations and personal knowledge can be easily exploited for building the data warehouse.

Nowadays, the actual business needs require enterprise information systems to have a great flexibility concerning the allowed business analysis and the treated data. Temporary alliances of enterprises, market analysis processes, the data availability on Internet push enterprises to quickly integrate unexpected data sources for their activities, and data sources populating the data warehouse may not be directly known and managed by the designers. For this reason, the reference scenario for data warehouse systems is deeply changed. In order to cope with this new scenario, we have classified four major critical requirements for effective ETL support:

1. **Automating extraction processes.** Designers can no longer rely on internal documents, comments and previous knowledge on the data source contents. Moreover, the manual exploration of the data source contents may be really time-consuming. Techniques for identifying the information held by the data sources and extracting the relevant data for populating the data warehouse are required.

2. **Automating transformation processes.** Data from different sources can not be homogeneous, i.e. different metrics can be used for expressing sim-
ilar values, synonyms can be used for representing the same values (and vice-versa the same value in different sources can describe a different concept), and values can be expressed with different granularity levels. Therefore, the data transformation processes are crucial for making attribute values from different data sources comparable: techniques to automatically perform data transformations are needed.

3. **Relaxing the transformation processes.** Deeply homogenizing transformations risk to flatten data warehouse structure and contents, thus allowing less accurate data analysis on such data. A balance should be pursued between ability of introducing new sources and preservation of structural and content homogeneity allowing accurate business intelligence processes.

4. **Speeding-up and easing ETL process execution.** Fast and simple ETL execution is a crucial competition factor when the sources populating the data warehouse are dynamic. Moreover, ETL tools should be able to manage different kinds of data sources, ranging from DBMS, to flat file, XML documents and spreadsheets.

In this paper we propose a tool for fulfilling requirements 1 and 2 above, i.e. for automating extraction and transformation processes. This tools couples and extends the MOMIS integration system and the RELEVANT system for the analysis of attribute values (see [5, 8] and related work section). By means of a semantic analysis coupled with clustering techniques, the tool performs two tasks: 1) it works at the schema level, identifying the parts of the schemata of the data sources which are related to the data warehouse, thus supporting the extraction process; 2) it works at the instance level, grouping the attribute values semantically related thus defining a transformation function for populating with “homogeneous” values the data warehouse.

The paper is organized as follows: the next section introduces some related work and an overview of both the previously developed MOMIS and RELEVANT systems. Section 3 and Section 4 describe our approach for supporting the automation of the extraction and the transformation processes. A simple running example extracted from a real application scenario helps the reader in understanding the approach. Finally, in Section 5, some conclusion and future work are sketched out.

2. **Related Work**

   Many approaches have been proposed for modeling ETL processes both at the logical and conceptual level. This plethora of approaches evidences that so
far the research community has not agreed upon the basic characteristics of ETL tools. Several methods, as in [11], propose extensions of UML to describe ETL processes. The advantage of those methods relies on the fact that they are based on a well-accepted, standard modeling language. Other approaches are based on ‘ad hoc’ techniques having the benefit of representing ETL processes without any restriction imposed by a previously defined, generic language. In [4], a set of basic operators on ETL processes, such as the Match, Merge, Rewrite, Invert, are introduced for the high level management of generic ETL processes. The Orchid approach [9] implements an abstract operator model (OHM) for representing data transformation operations independently of the specific ETL platform. The set of operator includes relational algebra operators (projection, selection, join, ...), operators for supporting nested structures, operators for managing the process as split (for processing the data by multiple subsequent operators) and other operators defined as subtype of the previous ones (e.g. Keygen that introduces and populates a new surrogate key column is a refinement of the project operator). Orchid implements a simplified version of the model introduced in [12]. In [17], a taxonomy of ETL activities based on the relationship of activity input to its output, in terms of both its schemata and the way it processes incoming tuples, is proposed. ETL activities are evaluated as unary processes, N-ary processes and routers & filters. Finally, in [16], the main ETL operations are analyzed and classified on the basis of their characteristics. A set of test suites for the problem of experimenting with ETL scenarios is proposed both for research purposes and for the comparison of the developed tools.

Recently, techniques exploiting ontologies and semantic web technologies to manage the heterogeneity of the data in the mapping and transformation processes have been proposed. In [14] and [13], an approach similar to the one proposed here is introduced: the authors exploit ontologies to formally and explicitly specify the semantics of the data source schemata as well as the data warehouse schema. In particular, in their approach the designer manually constructs a common ontology for representing the semantics of the sources. All the schema terms are annotated, i.e. manual relationships between ontology elements are defined, with respect to the ontology and the relationships in the ontology are exploited for determining the attributes/relations of the data sources where to extract information in order to populate the data warehouse. Differently from this approach, we propose a completely automatic process that exploits a common pre-existing reference ontology for annotating a meaning to each source element so that relationships and hierarchies among them are derived.

Besides research efforts, currently, there is a large variety of ETL tools avail-
able in the market (see [10] for a survey). All major database vendors provide ETL solutions bundling them with their DBMS [1, 3, 2].

In the following, the MOMIS and RELEVANT systems that are exploited and extended in this paper are introduced.

2.1. The MOMIS system at a glance

The Mediator Environment for Multiple Information Sources (MOMIS)\(^\text{2}\) [5] is a semiautomatic data integration system that follows a global-as-view approach to provide an intelligent access to heterogeneous, structured and semi-structured information sources. MOMIS aims at creating a conceptualization of the involved local sources by means of a reconciled and integrated Global Virtual View (GVV). The GVV generation process can be outlined as follows:

1. **Extraction of Local Source Schemata.** Wrappers acquire schemas of the involved local sources and logically convert them into ODL\(_{p3}\) (an extension of the ODL standard language - http://www.odmg.org);

2. **Local Source Annotation.** MOMIS recognizes that terms for describing schemas or structures in information sources hold exploitable semantics. In the local source annotation phase, the integration designer manually chooses the appropriate WordNet meaning(s) for each conceptual schema element provided by the wrapper;

3. **Common Thesaurus Generation.** Starting from the annotated local schemata, MOMIS constructs a Common Thesaurus that describes intra and inter-schema knowledge in the form of synonyms (SYN); broader terms/narrower terms (BT/NT); meronymy/ holonymy (RT). The Common Thesaurus is incrementally built by adding schema-derived relationships (automatic extraction of intra schema relationships from each schema separately), lexicon-derived relationships (inter-schema lexical relationships derived by the annotated sources and WordNet interaction), designer supplied relationships (specific domain knowledge capture) and inferred relationships (via description logics equivalence and subsumption computation);

4. **GVV generation.** Starting from the Common Thesaurus and the local schemata descriptions, MOMIS generates a global virtual view (GVV) consisting of a set of global classes GCs, plus mappings to connect the global attributes of each GC with the attributes of the local sources;

\(^\text{2}\)see http://www.dbgroup.unimo.it/Momis/ for more details and publications
5. GVV annotation. With the aim of making available the GVV to external users and applications, the GVV is semi-automatically annotated according to WordNet.

2.2. The RELEVANT prototype

RELEVANT [8] is based on the idea that analyzing an attribute domain, we may find values which may be clustered because strongly related. Providing a name to these clusters, i.e. the relevant values, we refer to a relevant value name which encompasses a set of values. More formally, given a class C and one of its attributes At, a relevant value for it, rvAt is a pair rvAt = ⟨rvnAt, valuesAt⟩. rvnAt is the name of the relevant value set, while valuesAt is the set of values referring to it.

RELEVANT is based on the following functional tasks:

1. Data pre-processing: the problem is to find an effective representation of the attribute values in a space, and to devise a suitable similarity function to be exploited by the clustering algorithm. After usual stemming, we build a binary representation of the attribute values, exploiting three different kinds of measures: 1) syntactic, mapping all the words of the attribute values in an abstract space, and defining a syntactic similarity function in such space; 2) dominance, introducing a sort of generalization relationship between values; and 3) lexical, which identifies semantically related values expressed with a different terminology;

2. Similarity Computation: two tasks are enabled: the selection of the metrics for computing the similarity between pairs of attribute values and the selection of the similarity measures to be used (syntactic, dominance, lexical or a combination of the three);

3. Clustering technique: this module implements some clustering algorithms to compute the set of relevant values on the basis of the selected similarity computation. The designer may choose between a classical clustering algorithm (generating partitions), and an overlapping clustering algorithm to compute values;

4. Name selection: The simplest way to detect a list of values candidates, i.e. the maximal values among values, is to use the Dominance similarity. The designer may select the most appropriate name among them;

5. Decision dashboard: the integration designer may interact with RELEVANT in two ways: with the simple or advanced mode. The simple mode uses
some default parameters and provides four sliders: the first to select the precision of the relevant values set. The slider ranges from rough, producing a small number of large relevant values where an attribute value may belong to different relevant values, to accurate, generating a large number of small relevant values each of them containing closely related attribute values. The second/third and fourth sliders allow the selection of the weight of the similarity computation method. In the advanced mode, the designer may set among about hundred different configurations and clustering thresholds;

6. Feedback: The system provides a set of standard quality measures to evaluate the quality of the clustering structure and supporting the user in the tuning activity.

3. Supporting the extraction process with MOMIS and RELEVANT

The extraction tool is based on the application of clustering techniques to the schema descriptions of the data warehouse and of the data sources to be added to the data warehouse. The result is a set of clusters, each one containing strongly-related schema elements. A process of cluster analysis allows the tool to generate mappings that represent relationships between the data warehouse schema and the data source schemata. Typically these are one-to-many mappings connecting one element from the data warehouse with several corresponding elements of the data sources.

In particular, for each data warehouse table (the target source), a set of mappings $m_{ij} = (t_{ai}, \{l_{aj}\})$ is defined, where $t_{ai}$ represents a data warehouse attribute and $l_{aj}$ a data source attribute (or a set of them). The union of the mappings $m_{ij}$ is called M.

Clustering techniques applied to literal descriptions, like the above depicted, are based on similarity measures and threshold levels that are typically domain dependent. Nevertheless, the selection of the most suitable similarity measure and threshold for a particular domain is a complex task: the tables and the attributes may not provide enough context or refer to different domains, thus making the selection a non-trivial and difficult to be automated process. Moreover, this choice may not be delegated to the user, since the skill required for doing it is generally not acquired by ETL users.

Our proposal addresses this issue with a semi-automatic approach where the user is required to supervise a process of extraction and enrichment of the semantics expressed by the descriptions of the schemata. The result of this task is a thesaurus of semantic relationships between the schema elements of the sources.
The knowledge in the thesaurus is then exploited for reducing the amount of possible customization parameters required by the clustering algorithm. The tool proposes two algorithms to be selected by the user (one computing a partition of the elements, the second one a set of overlapping clusters) where the user has to select the value of few coefficients that allow ranging the results from precise and well-selected to broad and approximate mappings. Since these coefficients have a direct and significant effect on the results, the user may easily evaluate his choices and in case refine the results with new selections. Moreover, some indexes measuring the semantic cohesion between the elements involved in a mapping have been implemented and are shown to the user supporting this task.

3.1. Functional architecture of the extraction tool

Figure 1 shows the functional architecture of the extraction tool, where the process is divided into three phases. The first phase concerns the semantic enrichment of the schema descriptions. This process is managed by exploiting the technique for data integration implemented in MOMIS. The result of this phase is a thesaurus of relationships between the schema thesaurus are then exploited by
an extension of RELEVANT for computing clusters of similar elements. Finally, in the third phase, the clusters are validated by the user with the support of quality indexes. The mappings between the data warehouse schema and the new data sources are provided by an analysis tool applied to the validated clusters.

3.2. Running example

The real scenario we refer to is an ongoing experiment within the LISEA lab project, funded by Italian Emilia Romagna region. It concerns the ETL process for the creation of a data warehouse for a commercial business intelligence application of the DataFlow company. DataFlow is a software company, leading the field of beverage and food logistics software, which developed a new application, called Bollicine Community business Intelligence (BCI). BCI is based on a predefined data warehouse which is automatically available for companies which already have the DataFlow Business management software. On the other hand, it needs ad-hoc procedures for data coming from other information systems. Among the DataFlow customers, there is an important Consortium of producers and distributors in the food and beverage area, made up of hundreds of companies, ranging from SMEs to multi-national corporations. The consortium wants to provide its members with the BCI solution, reaching two objectives: 1) analysing and planning of the enterprise market starting from its past data; 2) developing a performance benchmarking w.r.t. general indexes (KPI) obtained by aggregating data of all the members.

To reach this goal it is necessary to load all the data about the consortium members in the BCI data warehouse. We experimented our tool in this context: preliminary qualitative results show that the enterprises considerably save human resources. We will use this example throughout the paper to show some details of the tool operations.

Let us consider, for example, the Sales fact table of the BCI data warehouse, constituted of 3 analysis dimensions:

SALES(CODE, ARTICLE_CODE, BRANCH_CODE, DATE_MONTH, AMOUNT, QUANTITY, HECTOLITRES)

ARTICLE(ARTICLE_CODE, DESCRIPTION, CATEGORY_DESCRIPTION, SUBCATEGORY_DESCRIPTION)

BRANCH(BRANCH_CODE, DESCRIPTION)
3.3. Semantic enrichment

The semantic enrichment is performed by extending the techniques implemented in MOMIS. Three main activities are executed in this step: the extraction of the schema descriptions from the data sources, the annotation process and the creation of the thesaurus.

The schema descriptions of data sources are extracted by means of wrappers. A wrapper-based technology improves the tool flexibility and usability: the wrapper is responsible for providing a description of the source structure into the language internally used by the system: in our tool ODL43, an object-oriented language that is an extension of the standard ODL-OMG (in particular, a rich set of datatypes is added to better identify currency, datetime, unit of measure). Wrappers for several kinds of data sources have been developed (spreadsheets, xml documents, text files, JDBC databases, RDF documents, . . . ).
The annotation task consists of providing a well-known meaning to all the schema elements, by associating to each element the most suitable concept of a reference ontology / database (in our tool, the WordNet\(^3\) lexical database). The user may manually annotate the sources or refine the annotations suggested by the tool (see [7] where a probabilistic approach for annotating with respect to WordNet is proposed and [15] where a technique for automatically expanding acronyms and abbreviations in schema labels is described).

The knowledge engine is the component in charge of creating a thesaurus of relationships between the schema elements. Classes and attributes may be involved in three kinds of intra- and inter- source relationship: relationship connecting synonym (SYN), broader terms/narrower terms (BT/NT) and generic relationships (RT). The process for extracting relationships is extended from MOMIS and is based on structural analysis (where relationships are defined between elements in tables connected by foreign key, relationships between attributes in the same tables, . . . ), annotations (the relationships among the concepts in the referenced ontology / database are exploited for inferring relationships among the annotated elements) and Description Logics techniques.

**Example**

We exploited our tool for loading the data of three new companies into the data warehouse. The result is that correct mappings have been found, improving speed and result quality of the extraction process. Let us focus on the attributes FAMILY\_DESCRIPTION(S1), CATEGORY\_DESCRIPTION(S2), CLASS\_LABEL(S3) of the new sources in figure 2. The designer may set the tool to take into account only the syntactic similarity measure, thus obtaining two clusters, one made of the FAMILY\_DESCRIPTION(S1), CATEGORY\_DESCRIPTION(S2), CATEGORY\_DESCRIPTION(DW) attributes and the second one with only the CLASS\_LABEL(S3) attribute. Consequently the attribute CATEGORY\_DESCRIPTION(DW) is mapped with the corresponding attributes in S1 and S2. Since the attributes FAMILY\_DESCRIPTION(S1), CATEGORY\_DESCRIPTION(S2), CATEGORY\_DESCRIPTION(DW) are annotated with the same “description” concept in WordNet and CLASS\_LABEL(S3) is annotated with “label” that is a hyponym term of “description” in WordNet, the knowledge engine generates this set of SYN relationship in the thesaurus.

\[\text{ARTICLE.CATEGORY\_DESCRIPTION(DW) SYN MERCHANDISE.FAMILY\_DESCRIPTION(S1)}\]

\(^3\)http://wordnet.princeton.edu/
3.4. Generation of clusters

Clusters of related elements are computed by means of the technique implemented in RELEVANT, that has been extended by adding new similarity measures. In particular, clusters are created by exploiting three similarity measures:

1. syntactic similarity, which compares the alphabets used for describing the attribute values;
2. memberships, which measures the closeness of attributes belonging to the same table;
3. semantic similarity, which takes into account the thesaurus of relationships between classes and attributes.

Each similarity measure is internally represented with an affinity matrix, where a similarity value is computed for each attribute with respect to all other attributes according to the selected semantics. The combination of the values of each affinity matrix is parametric, thus allowing a setup phase where the user assigns an importance degree to each similarity measure. The application of a clustering algorithm (with a user-select threshold) generates clusters of similar elements.

Example

The relationships computed in the example of previous subsection may be exploited with the semantic similarity thus obtaining a unique cluster with all the attributes and consequently a set of mappings between the FAMILY_DESCRIPTION(DW) and the corresponding attributes of the new sources.

The process is performed for each target table, the defined mapping for the targeted table SALES are shown in table 1. As we can note, the AMOUNT attribute is mapped, in the data source S3, with two different attributes (COST and GAIN). Furthermore, for the datetime attributes the formats are formally specified (for example European or American) and the currency Euro or Dollar is indicated.

3.5. Cluster validations and mapping generations

Let us consider the case when we are adding a new source to the data warehouse. In the validation step, the obtained clusters are evaluated by the user and
in case transformed in mapping showing the elements of the data source related to the elements in the data warehouse.

The user is supported in the cluster evaluation by means of statistical and semantic indices automatically computed by the tool.

Statistical indices take into account the number of clusters computed for providing global measures of the quality of the final result. In the following $d$ will stand for the data warehouse before the insertion, $ds$ for the data warehouse after the insertion of source $s$. Then $nr_d$ is the number of clusters generated by the tool considering only $d$ as input (this value is typically the number of tables), $nr_s$ the number of clusters generated considering only the schema of the new source $s$ and $nr_{ds}$ the number of clusters generated considering both the data warehouse and the new source. We introduce some indices for supporting the user in evaluating the homogeneity of the data warehouse and the data sources on the whole. In particular, we assume that if the data source concerns a scenario similar to the one managed by the data warehouse, the application of the clustering tool to the data warehouse after the insertion of a new source will have two effects:

- it produces a number of clusters near to that of the data warehouse before the insertion,
- the clusters generated in the new source alone and in the data warehouse will be similar.

If either of the two conditions above do not hold, then the new source is in some way different from the data warehouse, and the user must decide if the insertion has to be accepted. We will consider three indicators for this evaluation:

<table>
<thead>
<tr>
<th>Table 1: Mappings for the SALES target table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SALES (DW)</strong></td>
</tr>
<tr>
<td>CODE</td>
</tr>
<tr>
<td>ARTICLE_CODE</td>
</tr>
<tr>
<td>BRANCH_CODE</td>
</tr>
<tr>
<td>DATE_MONTH</td>
</tr>
<tr>
<td>(European)</td>
</tr>
<tr>
<td>AMOUNT</td>
</tr>
<tr>
<td>(Euro)</td>
</tr>
<tr>
<td>QUANTITY</td>
</tr>
<tr>
<td>HECTOLITRES</td>
</tr>
</tbody>
</table>
• the before/after insertion cluster ratio \( ddsr = \frac{nr_d}{nr_{ds}} \)

• the source/populated warehouse cluster ratio \( sdsr = \frac{nr_s}{nr_{ds}} \)

• the source/populated warehouse Jaccard index; let us define as \( cs \) the clusters obtained from the new source, \( cds \) the clusters obtained from the data warehouse after insertion and \( pi(cds, cs) \) the projection of the clusters of the data warehouse on the new source (i.e. the clusters obtained excluding from \( cds \) the elements which are not included in the new source); the index measures the relatedness of a data source to the data warehouse by means of the Jaccard Coefficient\(^4\)

\[
JC_s = JC(s, pi(ds, s))
\]

Apart from the homogenous case mentioned above, when all the above indicators are 1, we can distinguish other interesting cases:

**smaller domain** in this case the new source is consistent with a subset of the data warehouse and the insertion can be done without any concern; this case is characterized by \( ddsr \) near 1, \( sdsr \) is less than 1 and \( JC_s \) is near 1;

**different domain** in this case the new source represents a domain significantly different from the data warehouse; this case is characterized by \( ddsr \) and \( JC_s \) significantly less than 1 (\( sdsr \) is not meaningful);

**extended domain** in this case the new source is partly consistent with the data warehouse, but introduces extensions; this case is characterized by \( ddsr \) and \( sdsr \) less than 1, and \( JC_s \) near 1.

Once the clusters are validated and approved by the user, mappings are automatically generated by analyzing the elements grouped in the clusters. The following cases are possible:

\(^4\)We recall that when we are comparing how similar are two partitioning schemes of a set we count the following numbers: \( f_{11} \) how many elements are in the same subset in the two schemes, \( f_{10} \) and \( f_{01} \) how many elements are in the same subset in the first scheme and in a different subset in the second, and vice-versa; then the Jaccard Coefficient is defined as \( f_{11}/(f_{01} + f_{10} + f_{11}) \) and is 1 is the two schemes are identical.
1. A cluster contains attributes from the data warehouse schema and the new data sources: each data warehouse element is mapped into all the data source elements in the cluster.

2. A cluster contains only attributes of the new data sources: this case is not exploited for the mapping generation, since it is due to the choice of a too selective clustering threshold.

3. A cluster contains only attributes of the data warehouse schema: no mappings are generated from this configuration. This kind of cluster may show that the data warehouse schema could be refined since there are different schema elements (the ones in the cluster) semantically related that may, in case, be fused into a unique element.

**Example**

Intuitively, let us consider the addition of source S3 w.r.t. the data warehouse. We can observe in table 1 that S3 attributes perfectly fit in one of the clusters of the data warehouse schema, than S3 is perfectly homogeneous to the data warehouse, and the number of clusters after the insertion will not change.

### 4. The Transformation Function Model

Transformation functions are implemented for homogenizing attribute values of different sources, thus allowing users to populate the data warehouse attributes with homogeneous values and to perform data analysis processes. These functions are typically exploited for solving data reconciliation issues, enabling the system to face very common data conflicts such as duplicated records, different representations of the same real world objects, different interpretations of values and different aggregations [14]. The OHM model proposed by the Orchid system [9] introduces some common transformation functions to be provided by an ETL system. According to that study, our tool proposes a transformation function model composed of the following functions:

- **RETRIEVE**: retrieves all the data values from a specific data source given as parameter.

- **PROJECT**: permits different operations involving data tables. In particular the following operations are supported:
  
  - *Column Split*: splits the content of a single column into multiple target columns (or a specific part of a local column to the output one);
– **Column Merge**: merges two or more local columns into a single target column;

– **Surrogate Key**: defines a unique record identifier (surrogate key) for each record of a fact table obtained with data coming from different sources. The surrogate key guarantees the preservation of the domain homogeneity, when the same primary key is used in different sources to represent different objects.

- **UNION**: performs the union of data records from two or more data sources.

- **JOIN**: permits the join among different local data sources obtaining object identification, i.e. different instantiations of the same real world object are identified [6].

- **CONVERT(c1,c2)**: converts all the values of a specified attribute from the pattern c1 to the pattern c2. Several patterns are predefined in the system:

  – **Date & Time**: date values may be represented in data sources with different formats. Therefore, the process for extracting date values from all the sources and merging them into a single unified attribute has to face reconciliation issues. We propose the function, `DATETIME(<TARGET ATTRIBUTE>, Local-Format, TargetFormat)` to automatically convert the data (and time) format of a source attribute to a target one. The function supports the ISO 8601 International Standard and most of the common DBMSs data and time formats.

  – **Currency**: converts the currency values retrieving from the web the current rate.

  – **Unit of measure**: converts the values from the source attribute to the target attribute by transforming the unit of measure.

- **FILTER(c)**: filters data records coming from a local data source on the basis of a specific boolean condition.

- **AGGREGATE(c)**: proposes an aggregation function (SUM, AVG, COUNT, ...) to the source attribute.

Several ETL systems and models (like the OHM model) propose a very poor set of transformation functions for string data-type, that are frequently based on merge
columns or small syntactic changes (i.e. upper and lower functions, stemming operation, regular expression rules). We propose a technique that provides a concise representation of a string domain attribute, by grouping the attribute values into semantically homogeneous clusters.

By means of this function (called *Semantic string categorization*), semantically related values of a chosen attribute in a data source and the correspondent values of a data warehouse are grouped in clusters, thus providing a semantic reconciliation of the attribute values. The transformation function is defined in a declarative way by the following syntax:

\[
\text{RELEVANT\_VALUE(<TARGET\_ATTRIBUTE>,<DATA\_SOURCE\_LIST>)}
\]

where \(<\text{TARGET\_ATTRIBUTE}>\) represents the attribute of the target schema whose values are grouped and \(<\text{DATA\_SOURCE\_LIST}>\) is the list of the attributes of the data sources to be clustered. The transformation function is based on the algorithm implemented in RELEVANT (see section 2.2) for computing the clusters. According to the RELEVANT algorithm, for each computed cluster a representative value is selected. Only this value, that we consider as the synthesis of all the values in the cluster, will populate the data warehouse and a temporary mapping table stored in a staging area will preserve the original values, allowing precise data analysis, if needed. The transformation function works accordingly to the following steps:

- **Attribute domains analysis.** *RELEVANT* is used for evaluating if the attribute values of the data sources and the data warehouse are compatible, i.e. they contain semantically similar values. The similarity evaluation is the one reported in section 3.5.

- If the domains are compatible, it is possible to replace the attribute values of a data source with the relevant values obtained by the data and the data warehouse sources. By means of this function, the values of such attribute, typically an analysis dimension, will consist of homogeneous values thus allowing OLAP analysis (i.e. drill down) on them.

- If the domains are not compatible, the user may choose to transform and load into the data warehouse attribute only the compatible values (i.e. the values belonging clusters with values from the data warehouse attribute). In this case, a temporary mapping table is built for preserving the possibility of more accurate data analysis. The advantage of such solution is a reduced cardinality of values in the data warehouse and consequently a more synthetic representation of the object.
Algorithm 1: **Transformation Function Identification (TFI)**

**Input:** all mappings $m_{ij}(t_a, \{l_{aj}\}) \in M$

**Output:** the set of transformation functions proposed for $m_{ij}(t_a, \{l_{aj}\})$

1. **foreach** table $\in$ target source
2. **foreach** $m_{ij}(t_a, \{l_{aj}\})$
   3. /* mapping level */
   4. if cardinality($\{l_{aj}\}$) > 1
   5. MERGE($\{l_{aj}\}$)
   6. if ($l_{aj}$) = any $(l_{ap}) \in m_{pq}(t_{ap}, \{l_{ap}\}), \forall m_{pq} \in M$
   7. FILTER($l_{aj}$)
   8. if datatype($t_a$) is a refinement of datatype($l_{aj}$)
   9. CONVERT($t_a, l_{aj}$)
10. if (annotation($l_{aj}$) contains {'code', 'identifier', 'ID'}) or ($l_{aj}$) is PK or ($l_{aj}$) is FK
11. SURROGATE_KEY($l_{aj}$)
12. /* semantic annotation level */
13. if annotation($l_{aj}$) is an hypernym of annotation($t_a$) or annotation($l_{aj}$) and annotation($t_a$) are siblings
14. CONVERT($l_{aj}$)
15. if annotation($l_{aj}$) is an hyponym of annotation($t_a$)
16. AGGREGATE($l_{aj}$)
17. /* domain compatible level */
18. **foreach** datatype($\{l_{aj}\}$) = string and domain($\{l_{aj}\}$), domain($\{m_{ij}\}$) are compatible
19. RELEVANT_VALUE($t_a, \{l_{aj}\}$)

### 4.1. Automatically selection of transformation functions

Taking into account the set of functions proposed by our Transformation Function Model (see the previous section), we introduce a method that proposes the suitable transformation function to support the user during the transformation phase of the ETL process. Our approach takes into account the results of the semantic enrichment process performed by the extraction tool (see section 3) and in particular the semantic annotations, the data types and the analysis of data instances to infer incoherence and conflicts over the data mappings.

The process is described by the Transformation Function Identification (TFI)
algorithm (algorithm 1), that takes as input all the mappings between the data source and the data warehouse attributes and the thesaurus generated by the extraction process applying on it a reasoning process. The TFI algorithm works at three different levels: mapping, semantic and domain compatible level.

- **mapping level**: for each target class \(i\) in the data warehouse, the attribute values to be merged and filtered are identified. In addition, the \(t_a\) and \(l_a\) types are checked and the CONVERT function is in case proposed. Finally, for Primary Key and Foreign Key attributes or attributes annotated with terms like ‘code’, ‘identifier’ or ‘ID’, the SURROGATE KEY function is proposed.

- **semantic level**: we consider the semantic relationships holding among the annotations of the attributes, if a hypernym or a sibling relationship holds between data warehouse and local attributes, the CONVERT function is proposed. On the opposite, if the local attribute is an hyponym of the target level, an AGGREGATE function is proposed.

- **domain compatible level**: for mappings involving string datatype attributes, we apply the RELEVANT VALUE function to the set of compatible values of the local and the data warehouse attributes.

The algorithm proposes a function for each element of the mapping table: these functions are described in a declarative way (name and parameters) and can directly executed by our system.

4.2. Running example

Table 2 shows the transformations proposed by our tool for the SALES target table, according to the TFI algorithm. At the mapping level, for key attributes (`CODE`, `ARTICLE_CODE`, `BRANCH_CODE`), the SURROGATE KEY function is proposed. Concerning the `DATE_MONTH` attribute, the CONVERT function is proposed, since in S2 the American datetime format is used while in the target schema the European one is present. Concerning the `AMOUNT`, in S2 a CONVERT (from Dollar to Euro) function is proposed, while in S3 the MERGE function unifies the `COST` and `GAIN` attributes. Since the `QUANTITY` attribute in S3 refers to two different mappings, the FILTER function is proposed. At the semantic level, since ‘capacity’ is an hypernym of ‘hectoliter’, a CONVERT function is proposed.
Finally, at the domain compatible level, let us consider the BCI table describing the articles sold by companies:

```
ARTICLE(CODE, DESCR, CATEGORY_DESCRIPTION, SUBCATEGORY_DESCRIPTION)
```

where `CATEGORY_DESCRIPTION` is a dimension for BI analysis. Three sources are involved in the data warehouse with different string values to describe similar article categories:

- Source: #1 Attribute: `FAMILY_DESCRIPTION`
  Values: {NOT DEFINED, WATER, BEER, WINE, SOFT DRINK, FOOD}
- Source: #2 Attribute: `CATEGORY_DESCRIPTION`
  Values: {HARD LIQUOR, BOTTLE WINE, NOT DEFINED, JUICE DRINK, MINERAL WATER, BOTTLE BEER, ALCOHOLIC BEVERAGE}
- Source: #3 Attribute: `CLASS_LABEL`
  Values: {NOT DEFINED, MINERAL WATER, BEER, WINE, ALCOHOLIC DRINK, FOOD, BOOZE, FOOD CATERING}

The system proposes the application of the following function:

```
RELEVANT_VALUE(ARTICLE.CATEGORY_DESCRIPTION, S1.FAMILY_DESCRIPTION, S3.CLASS_LABEL, S2.CATEGORY_DESCRIPTION).
```

The result is a set of 7 clusters, each one with a representative name. Such values are loaded as values of the data warehouse attribute instead of the 22 original values:

RV1: NOT DEFINED {NOT DEFINED, NOT DEFINED, NOT DEFINED}
RV2: WATER {MINERAL WATER, WATER}
The name assigned to each cluster is obtained by using the order of the sources in the function call: the result is a SQL query based on the COALESCE function and that is applied to the clusters. Concerning the proposed example, the executed function is the following:

```
UPDATE CLUSTER_ARTICLE_CATEGORY_DESCRIPTION
SET TARGET = COALESCE
((SELECT MIN(CL1.CATEGORY_DESCRIPTION)
FROM CLUSTER_ARTICLE_CATEGORY_DESCRIPTION AS CL1
WHERE CHARINDEX(RTRIM(LTRIM(CL1.CATEGORY_DESCRIPTION)),
CLUSTER_ARTICLE_CATEGORY_DESCRIPTION.CLUSTER) > 0 AND
CL1.SOURCE = S1),
(SELECT MIN(CL1.CATEGORY_DESCRIPTION)
FROM CLUSTER_ARTICLE_CATEGORY_DESCRIPTION AS CL1
WHERE CHARINDEX(RTRIM(LTRIM(CL1.CATEGORY_DESCRIPTION)),
CLUSTER_ARTICLE_CATEGORY_DESCRIPTION.CLUSTER) > 0 AND
CL1.SOURCE = S3),
(SELECT MIN(CL1.CATEGORY_DESCRIPTION)
FROM CLUSTER_ARTICLE_CATEGORY_DESCRIPTION AS CL1
WHERE CHARINDEX(RTRIM(LTRIM(CL1.CATEGORY_DESCRIPTION)),
CLUSTER_ARTICLE_CATEGORY_DESCRIPTION.CLUSTER) > 0 AND
CL1.SOURCE = S2))
```

Figure 3 shows a fragment of the data warehouse ARTICLE table, where the original tuples are shown on the top, and the transformed ones on the bottom. The original values are not removed by the system, but stored in a temporary table for being exploited in detailed analysis concerning the values of the data warehouse attributes.

The algorithm calculates a transformation function for each element of the mapping table: in the example, over the 85% of these functions correspond to those manually proposed by a data warehouse designer.
5. Conclusion and future work

In this paper we proposed to couple and extend our previous research on data integration and data analysis for creating an ETL tool. In particular, we
focused our work on the extraction phase, by implementing a technique that semi-
automatically defines mappings between a data warehouse schema and a new data
source, and on the transformation phase, by proposing a new function based on
relevant values, particularly useful for supporting drill down operations. We ex-
perimented our approach on a real scenario, thus obtaining qualitative results on
the effectiveness of the approach.

Future work will be addressed on identifying a benchmark and a set of mea-
sures in order to perform a complete technique evaluation.

3.ibm.com/software/data/db2/datawarehouse/. IBM.


an integrated ontology. IEEE Internet Computing, pages 42–51, Sep-Oct
2003.

Mbinkeu. Getting through the THALIA benchmark with MOMIS. In Inter-

integration systems. In R. Meersman, Z. Tari, and P. Herrero, editors, OTM
Workshops (1), volume 4805 of Lecture Notes in Computer Science, pages

attribute values for improved search. IEEE Internet Computing, 11(5):26–


