Ontology-guided Reference Data Alignment in Information Integration Projects

Sushain Pandit
Information Management, IBM, Austin, TX, United States., sushain.pandit@us.ibm.com

Martin Oberhofer
Information Management, IBM, Austin, TX, United States., martino@de.ibm.com

Ivan Milman
Information Management, IBM, Austin, TX, United States., imilman@us.ibm.com

Ontology-guided Reference Data Alignment in Information Integration Projects

Sushain Pandit
IBM
sushain.pandit@us.ibm.com

Martin Oberhofer
IBM
martino@de.ibm.com

Ivan Milman
IBM
imilman@us.ibm.com

ABSTRACT

One of the hard problems in information integration projects (harmonizing data from various legacy sources into one or more targets) is the appropriate alignment of reference data values across systems. Without this alignment, the process of loading records into the target systems might fail because the target might reject any record with an unknown reference data value or different underlying data semantics. Today, detecting reference data tables and determining the relative alignment between a source and a target is largely manual, cumbersome, error-prone and costly. We propose a novel ontology-guided approach to detect reference data tables and their relative alignment across source/target systems to enable semi-automated creation of translation tables.

Keywords

Reference data, Information Management, Discovery, Ontology, Master Data Management

BACKGROUND AND INTRODUCTION

Information integration projects involve loading and harmonizing data from various source systems into one or more target systems. A typical large-scale information integration project is the creation of a Master Data Management (MDM) system (Dreibelbis, Hechler, Milman, Oberhofer, Run and Wolfson, 2008; Berson and Dubov, 2010) - other examples include SAP application consolidation as well as data warehousing (DW). Although MDM systems naturally emphasize managing master data, a related category of data, called *reference data*, is used to define aspects of the master data captured in these solutions. Examples of reference data include attributes like *gender, status codes, employee types, and states/province codes.* The range of domain values and descriptions for a given reference data attribute usually resides within specialized tables, known as *lookup, code, check* or *domain tables*. These tables, often large in number, contain code values and descriptions that tend to change less frequently over time as compared to master data tables. Reference data can also be differentiated from metadata (Narayanan, Oberhofer and Pandit, 2012) in that the latter describes structure of an entity whereas former describes only the permissible range of values for an attribute of an entity. Authors in (Chisholm, 2000; Fryman, Inmon and O'Neil, 2007) put various different categorizations around these data categories into perspective.

A typical MDM solution implementation has two major phases, both of which require integration of reference data. The first phase is *master data integration* and comprises MDM system installation and configuration, followed by loading of harmonized master data from a number of heterogeneous source systems. The second one, known as *master data distribution*, comprises maintenance and delivery of high-quality master data for a number of consumers. Although the overall solution architecture for these phases can vary (Dreibelbis, et al., 2008; Fot, Mandelstein, Milman, Oberhofer and Pandit, 2011), the high-level architecture shown in Figure 1 captures the core concepts. In addition, the figure also describes the sequence of major steps for the first phase. The data integration process begins when data from multiple sources (S_i) is extracted into a staging (STG) area, which has a sub-area corresponding to each source data model. The second step is the application of data profiling to extracted source data in STG, which identifies data quality issues needed to be fixed before the data is loaded into a target system. A comprehensive classification of data quality problems can be found in (Leser and Naumann, 2007).

Profiling may provide two additional specific services, viz. (i) domain analysis to check if a column contains values only permitted by a domain (set of reference values), which presumes that the domain values are known and (ii) transformation discovery to see if two tables are related, which does not do any transformation or linking of reference data, just of related columns between tables.

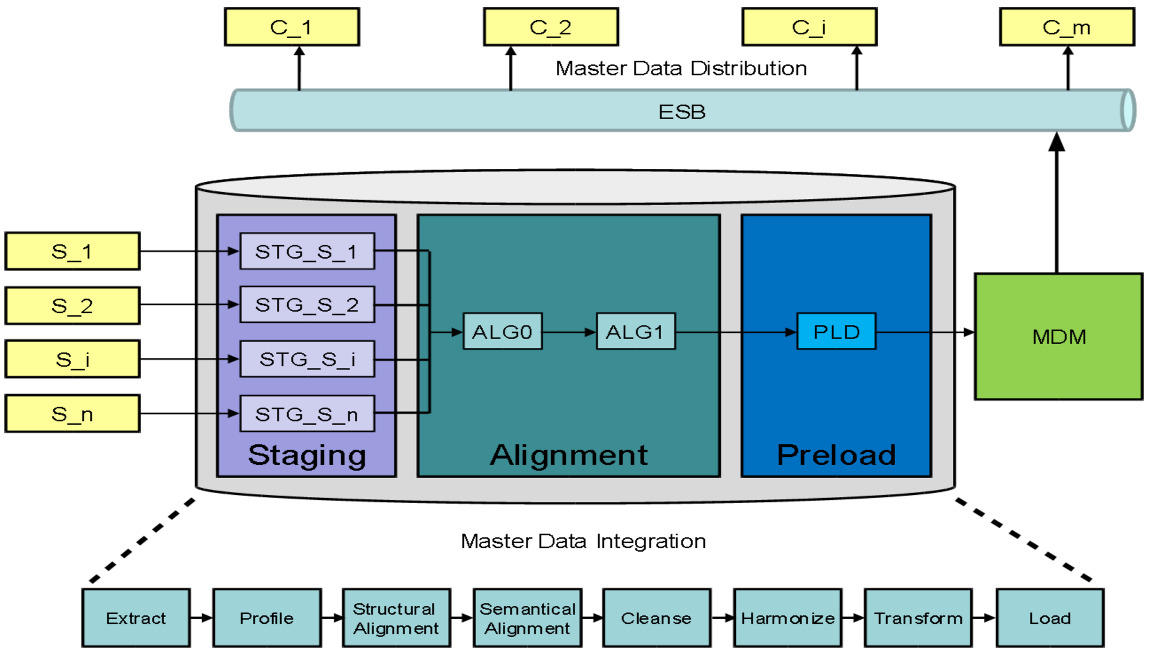
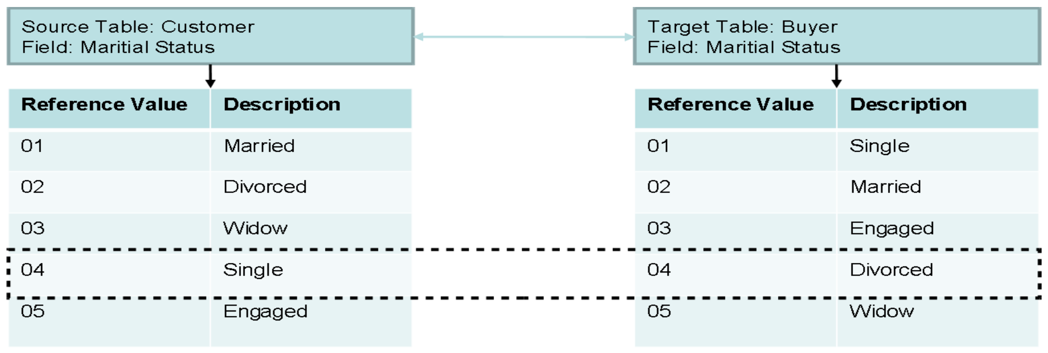


Figure 1: Conceptual Architecture for Master Data Integration and Distribution

The next step, called structural alignment, utilizes the alignment (ALG) area to align multiple different source data models into a common data model. The alignment (ALG) area is divided into two parts, viz. ALG0 and ALG1, both of which have the same data model. The key design principle is to ensure logical closeness between ALG and target system data models, adjusted in a way that all records across all data sources can be accommodated. E.g., if a source has a field representing *street* of 150 characters length and the corresponding field in the target has only 120 characters, the street field in ALG areas would be 150 characters to accommodate all source records. The next step is *semantic alignment*, which presents the key issues that we address in this paper (described in the next section). For brevity, the remaining steps, viz. data cleansing, data harmonization, transformation and load are omitted. An in-depth discussion on the major steps of data integration process can be found in (Fryman, et al., 2007; Godinez, Hechler, Koenig, Lockwood, Oberhofer and Schroeck, 2010). We now present the reference data translation problem in the semantic alignment step.

The Reference Data Translation Problem

Semantic alignment is applied when data is processed from ALG0 to ALG1 (refer to Figure 1). This step introduces the reference data translation problem as illustrated in Figures 2 and 3. More specifically, for each field within a table belonging to some source, if the field's domain values are backed by a reference data (or lookup) table, then those values are replaced by the corresponding reference values from the respective lookup table for the equivalent target system field. The reference data translation problem comes in two flavors. The first case is where the source and target systems have the same code value but with different semantics on the description (dotted box/lines in Figure 2). The second case is where the source and target systems have different code value sets for the same reference data domain. As shown in Figure 3, source system 2 uses the value “S” for an instance of the attribute Marital State, while the target system 1 uses the value of ‘01’ to represent the same concept. In either case, without replacing the reference data values from the source MDM system with their semantic equivalent in the target, the semantic integrity of the records during data integration process can not be guaranteed. The mechanism for this replacement is a translation table (refer Figure 3), which defines the basis for the rules to govern the replacement of reference values. Replacement rules must be defined per unique pair of source and target systems. Successful completion of this step helps all source records from multiple sources to be available in a common structure, sharing a common semantics and in turn, enabling common data cleansing logic to be applied across all sources.



Source Table: Customer
Field: Maritial Status

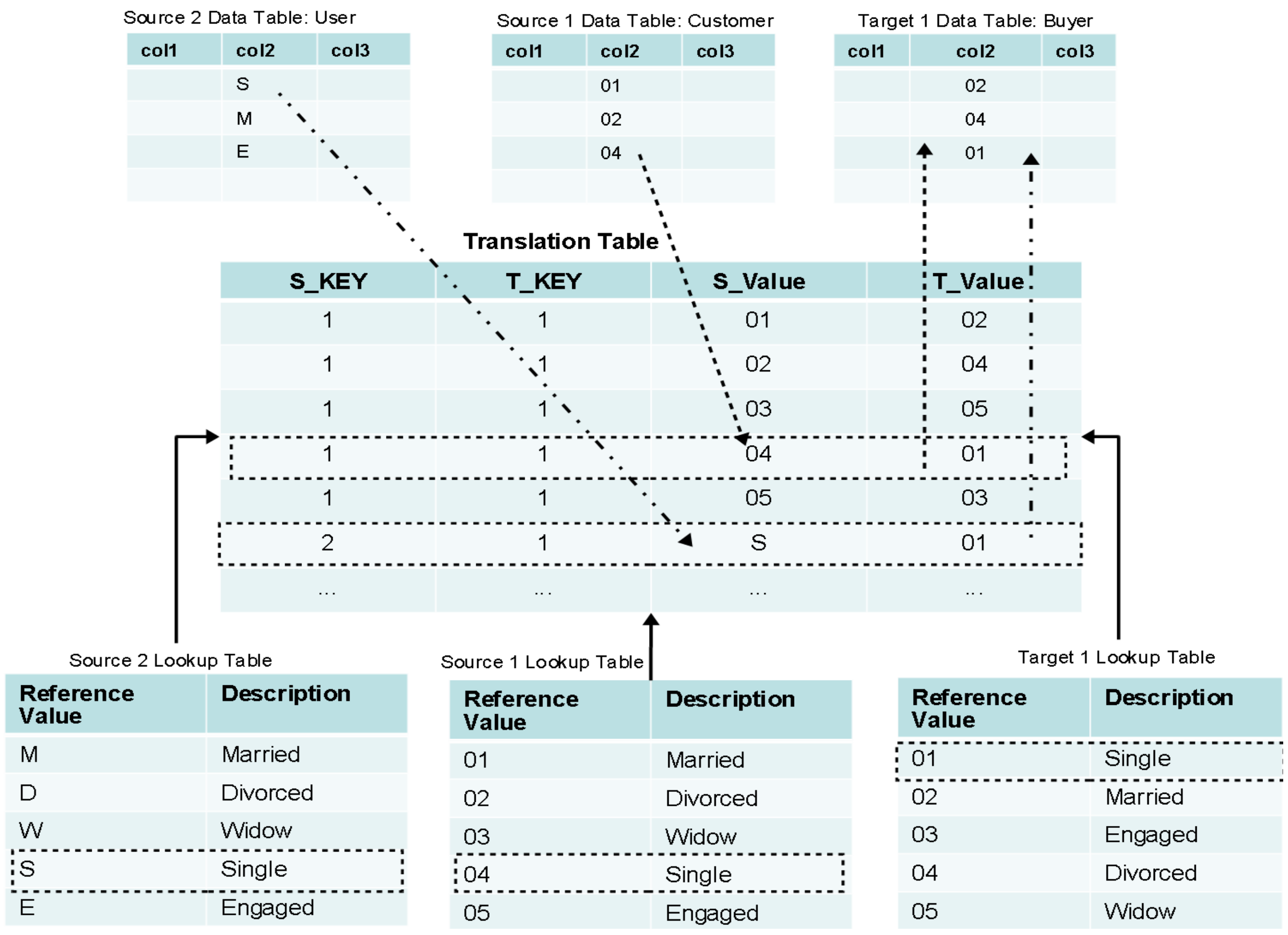
Reference Value	Description
01	Married
02	Divorced
03	Widow
04	Single
05	Engaged

Target Table: Buyer
Field: Maritial Status

Reference Value	Description
01	Single
02	Married
03	Engaged
04	Divorced
05	Widow

Figure 2: Reference Data Translation Problem

It may be observed that in the second MDM phase (master data distribution), the master data from the MDM system gets distributed to a large number of consuming applications (C_i) as shown in Figure 1. Since the consuming applications usually have different reference data sets than the MDM systems, we are faced with issues similar to the semantic alignment step described above. If the consuming application were a source during the master data integration phase, part of the solution could be re-used appropriately.



Source 2 Data Table: User

col1	col2	col3
	S	
	M	
	E	

Source 1 Data Table: Customer

col1	col2	col3
	01	
	02	
	04	

Target 1 Data Table: Buyer

col1	col2	col3
	02	
	04	
	01	

Translation Table

S_KEY	T_KEY	S_Value	T_Value
1	1	01	02
1	1	02	04
1	1	03	05
1	1	04	01
1	1	05	03
2	1	S	01
…	…	…	…

Source 2 Lookup Table

Reference Value	Description
M	Married
D	Divorced
W	Widow
S	Single
E	Engaged

Source 1 Lookup Table

Reference Value	Description
01	Married
02	Divorced
03	Widow
04	Single
05	Engaged

Target 1 Lookup Table

Reference Value	Description
01	Single
02	Married
03	Engaged
04	Divorced
05	Widow

Figure 3: Applying Translation Tables during Data Migration

Overall, the reference data translation problem for the semantic alignment of data with respect to the described architecture can be summarized into three distinct tasks:

1. Reference data identification that occurs in the staging area per source/target data models, with the objective of identifying all lookup tables across all the source and target systems.
2. Discovery of matching source and target reference data table to derive pairs of tables that require translation.
3. Construction of translation tables used by the semantic alignment step. To build the translation tables, we need to use:
    1. The lookup tables with all records extracted into STG from the sources
    2. Manually creating mapping rules that solve the reference data translation problem as part of the semantic alignment
    3. The lookup tables with all records extracted from the targets into PLD

Now, we perform a brief (but necessary) comparison with existing approaches to schema matching. Schema matching is the identification of semantically correlated attributes in tables present in two or more systems. Thus, the part of the reference data translation problem, where corresponding lookup tables and attributes in at least one source and target system need to be correlated with each other, could be considered a schema matching problem. A classification of schema matching techniques can be found in (Rahm and Bernstein, 2001; Bellahsene, Bonifati and Rahm, 2011). The challenges with schema matching when applied to this domain, are two-fold. Firstly, the techniques based on name similarities and data types of attributes does not work well in cases where source and target system are completely different (e.g, an MDM system using employee type as a character ('F','P') and a SAP system using employee type as a string ("FULL-TIME", "PART-TIME")). Secondly, instance-level matching techniques are not very useful either because two systems could use the same type (e.g, integer values), however the actual values may differ (e.g, one system may use (1,2) to represent full-time and part time employees, while another might use (0,1)).

Solving the reference data translation problem as part of the semantic alignment step is challenging because[1]:

- Data migrations projects often need to extract, transform and load (ETL) data from several dozens to several hundreds of source systems. The source system may comprise of packaged (e.g., SAP R/3, Siebel) or custom built applications using multiple programming languages and database technologies. In such projects, the same entity (e.g., Customer) is often stored in various different source systems with a different representation in each, and in particular, can include different values in the lookup tables for the same attribute.
- Even if the same application is deployed multiple times (by a business unit or country), due to possible application customization, the value sets in the same lookup table may differ across the application instances. E.g., the application owner in one business unit might use the default reference value set after installation of the application, whereas another application owner might decide to replace the default value set with a reference value set from a third-party (e.g. ISO code sets). In a large enterprise, for something relatively straightforward like country code, spread across 20 instances of one particular application, several thousand different country code values may be found. In addition, the target system might have a different set of reference values in the lookup table as well.
- The problem has to be tackled per pair of source and target system, and as noted, this can be a large number of systems.
- Business entities usually have several dozen to a few hundred attributes, with a measurable percentage being backed by lookup tables.
- Most legacy source systems do not have documentation on the data models, particularly reference data. This presents a two-fold problem. First, many applications have thousands to tens-of-thousands of tables in a database. E.g., older versions of an SAP R/3 System can have about 25,000 tables with more then 237,000 attributes (Bauckmann, Leser and Naumann, 2010). Consequently, it's not easy to determine the tables containing reference data. Second, even if it is known that a table stores reference data, without documentation, it is difficult to manually identify the domain for each lookup table.
- The construction of translation tables is completely manual and time-consuming, making it a costly exercise. The paper from (Li, Liu and Zhang, 2003) cites a project, where mere matching of attributes across tables would've been worth 12

[1] Examples in the list are drawn from authors' field experience

person years, if done manually. Since the reference data translation problem comprises a schema matching sub-problem, doing it manually is indeed hard.

For these reasons, the discovery of reference data in source and target systems for hundreds of reference data tables and creation of the translation tables that are required for successful data integration and distribution is a non-trivial problem.

At the same time, on the other side of information spectrum, there have been significant advances in capturing domain semantics within information landscapes in the form of ontologies (Simperl, Mochol and Burger, 2010). This presents an interesting opportunity to tackle the problem of reference data translation by exploiting the semantics inherent in such domain ontologies.

Against this background and problem description, we present an ontology-based method to discover the lookup tables in the source and target systems and then, match lookup tables from the sources with their counterparts in the target systems. After doing this semantic alignment, we suggest ways to auto-generate the translation tables to the extent possible.

The rest of this paper is organized as follows. Next section covers the preliminaries. Subsequent sections describe our approach to solving the reference data alignment problem in detail and present a discussion on the proposed approach. Last section concludes with a summary.

PRELIMINARIES

We make use of the following notation and definitions in this paper:

- $X = \{x_i\}$, denotes a set of code values
- $D = \{d_i\}$, denotes a set of descriptions corresponding to X
- $v_i = \{x_i, d_i\}$ denotes a reference data element
- $R = \{v_i\}$, denotes a reference data set or a lookup table
- T_s denotes the set of all candidate lookup tables (see below) in source systems.
- T_t denotes the set of all candidate lookup tables (see below) in target systems.

Candidate lookup tables are discovered by looking through all the existing tables in the systems and applying some known heuristics to determine whether each qualifies as a lookup table or not. These heuristics are discussed in the next section.

Definition 1: (Domain Ontology) A domain ontology is encoded by a 5-tuple *DOI* = {L, C, Y, h, f} where Y is a set of instances in the domain ontology, C is a set of concepts defined over Y, and L is a set of relations defined over Y **x** Y (meaning each element in L is a relationship between a pair of instances from Y), h is a function that takes as input, an instance $y \in Y$ and returns a set of concepts in C that contain the instance y, and f is a function that takes as input, an ordered pair of instances $\{y_1, y_2\}$ and returns the set of all relations in L that contain $\{y_1, y_2\}$. (Pandit and Honavar, 2010).

For a concept $c \in C$ captured in a domain ontology *DOI*, we define a value-partition, p_c, to be a set of concepts k_i, such that $c = \bigcup k_i$ and $\bigcap k_i = \emptyset$. This divides the concept c into k_i partitions. (Ontology Engineering and Patterns Task Force, 2005).

Definition 2: (Maximal Value Partition Concept) Given a set of descriptions D and a domain ontology *DOI*, we define a maximal value partition concept with respect to the set D as the concept $c \in C$, such that the value partition p_c contains maximum number of concepts with labels matching the descriptions from the set D.

Next, we now describe our approach in detail.

APPROACH

Our working hypothesis is that there is a fundamental similarity between code tables and value partitions in that both represent a set of related elements. We first identify lookup tables in source and target systems as well as assign them an abstract meaning by computing maximal value partition concept in *DOI* for each candidate lookup table present in those systems. Presence of a value partition for a table implies the candidacy of that table to be of type lookup and maximal value partition concept represents a domain abstraction for the type of values that the lookup table holds. Further, we make use of these identified concepts (maximal value partition) to further explore the ontology in order to discover possible semantic relationships. Existence of such relationships suggests a corresponding correlation between the lookup tables that these concepts are representing as domain abstractions.

Discovering Candidate Lookup Tables

Typically, the source and target systems comprise of tens of thousands of tables. As a result, running any sort of algorithm on every single table to determine if it is a reference table is not very efficient nor desirable. In order to reduce this search space, we make use of the fact that lookup tables comprise of relatively small number of columns and rows. As a result, the ratio of the number of rows (and/or columns) in the candidate table to the maximum number of rows (and/or columns) found across all the tables in the source and target systems, would be very small ($\rightarrow 0$). Thus, as a pre-processing step, by performing schema discovery for each table present in source and target systems and computing these ratios, we are able to rule out a number of tables and arrive at sets of candidate lookup tables, T_s and T_t, for source and target systems respectively.

Identifying lookup tables and computing a domain abstraction

Procedure P_1 consumes a domain ontology, a candidate lookup table, and tries to find a maximal value partition concept, the existence of which automatically identifies the table as a lookup table and the concept itself gives a domain abstraction for the lookup table.

```
Procedure P_1
1. Input domain ontology DOI and a reference data set (lookup table) $R = \{\{x_i, d_i\}\}$
2. Calculate the maximal value partition concept with respect to the set of descriptions $\{d_i\}$ as follows:
      We assume OWL-Lite (least expressive sub-language of OWL, the Web Ontology Language used for authoring
      ontologies) conventions for ontology DOI. There exists a corresponding transformation to RDF graph triples
      (Web Ontology Working Group, 2004).
   a) If $\{Q_i\}$ have been computed previously, skip to (b). Otherwise, traverse Ontology DOI along subclass
      relationships, starting at an arbitrary concept $c \in C$ and do until all concepts have been visited:
      i.   Expand all the concepts, $\{k_j\}$, that are directly connected to c
      ii.  Put all the elements in $\{k_j\}$ to a set $Q_c$, iff the following conditions hold:
           • $c = \bigcup k_j$ and $\bigcap k_j = \varnothing$
      iii. Mark c as visited
      iv.  Pick one of the expanded (unmarked) concepts and repeat
   b) Sequentially compare the elements in $\{d_i\}$ with each set in $\{Q_j\}$, and find out the set $Q_{max}$, $max \in C$, containing
      maximum number of elements from the set $\{d_i\}$
3. If no match is found, return null, else return the class concept $max \in C$, as the maximal value partition concept
```

Discovering Semantic Associations and Determining Source/Target Lookup Table Match

Procedure P_2 takes *DOI* and two instances as input and tries to find a semantic path between them using a series of semantic queries. The procedure can be understood by referring to the example scenario in Figure 4 where *System_A_cc_val* and *SystemB_cc_val* are two instances defined in a domain ontology. In this case, the procedure starts from *System_A_cc_val* node and tries to find a sequence of triple patterns such that for the first triple, one of the participating nodes is *System_A_cc_val* and in the final triple, one of the participating nodes has been annotated as a *target* node. In this case, it finds the highlighted path to *System_B_cc_val* (which is part of the target system) and returns it as the result, along with the target node. This path can be represented in RDF/XML notation as a series of triples as follows:

i.	http://uri#SystemA	http://uri#uses_code_value	http://uri#System_A_cc_val
ii.	http://uri#SystemA	http://uri#instanceOf	http://uri#System_en
iii.	http://uri#System_en	http://uri#subclass	http://uri#System

iv.	http://uri#System_fr	http://uri#subclass	http://uri#System
v.	http://uri#SystemB	http://uri#instanceOf	http://uri#System_fr
vi.	http://uri#System_B_cc_val	http://uri#uses_code_value	http://uri#SystemA

For this work, we do not deal with directionality of the semantic associations and thus, we simply look for triple patterns where a resource is common in at least the subject or the object.

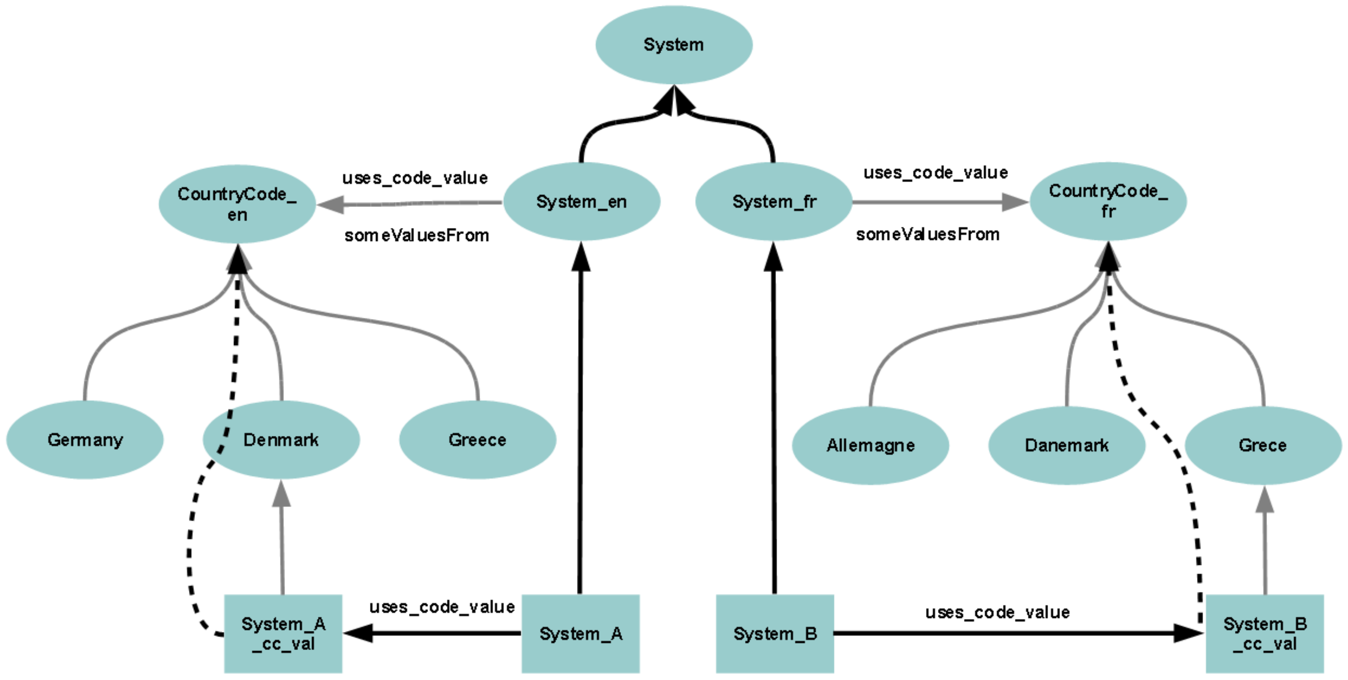


Figure 4: Relationship discovery for Source/Target Lookup Table Match

<u>Procedure P_2</u>

```
1. Input domain ontology *DOI* and instance y
2. Perform a search on the DOI graph to determine a path P of semantic associations from y to a target node as follows:
    a) Find a triple pattern $t_1$: <s, p, o> such that $t_1$:s = y OR $t_1$:o = y
    b) Add $t_1$ to queue, Q and Set found flag to false
    c) Do while Q not null:
        i. Eject triple $t_i$ from Q and mark $t_i$ as visited
            • If $t_i$:s hasAnnotation 'target' OR $t_i$:o hasAnnotation 'target', set found flag and break out of the loop
            • For all triples $t_j$: <s, p, o> such that $t_j$:s = $t_i$:s OR $t_j$:o = $t_i$:o OR $t_j$:s = $t_i$:o OR $t_j$:o = $t_i$:s :
                Add $t_i$ to Q
3. If found is false, return null. Otherwise, return the node (subject or object) from the triple $t_i$ containing the annotation 'target'
```

Composite Reference Data Alignment Framework

In this section, we combine the above described procedures in a composite framework to solve the reference data alignment problem. Procedure *main* takes as input T_s, T_t, *DOI* and determines semantic matches (if existing) between candidate tables in T_s and T_t. These matches are suggested to the user and finalized or rejected based on feedback. Each finalization becomes a mapping between a pair of candidate reference tables, in effect aligning them semantically with respect to each other. As a final step, the procedure builds the translation tables from the data values contained in the semantically aligned reference data tables.

Procedure main

1. Input set $T = T_s \cup T_t$, of all candidate lookup tables in the source and target systems and the domain ontology *DOI*
2. For each reference data set (lookup table) $R_m = \{\{x_i, d_i\}\}$ present in the set T, compute the maximal value partition concept n_m with respect to the sets $\{d_i\}$ by invoking procedure P_1 on parameters *DOI* and R_m
3. Store a reference (to be used later) from each maximal value concept n_m to the corresponding reference data set R_m
4. For each n_m, do:
    a) Find all the instances y_{mj} ($\in$ Y) of the concept n_m by referring the assertions present in *DOI*
    b) Annotate each instance y_{mj} with an annotation '*Source*' or '*Target*' based on whether the corresponding concept represents a domain abstraction for a source or a target table
5. For each instance y_{mj} found above such that y_{mj} has annotation '*Source*', do
    a) Invoke P_2 on parameters DOI and y_{mj}, and compute a related target instance y_{rel} (annotated with '*Target*')
    b) Suggest y_{rel} to the user (optionally with the path from y_{mj} to y_{rel}) to finalize or reject the discovered mapping
    c) Corresponding to each finalized mapping, determine the corresponding reference data sets, R_{m1} and R_{m2} participating in that mapping (by referring to the stored reference from maximal value partition concept to corresponding reference data set)
    d) Perform alignment between the data values of R_{m1} and R_{m2} (to the extent possible) to build translation tables as follows:
        - Compute the structure of the translation table, generate and execute create table statements in the migration database
        - Determine the language settings in the source and target system
        - Run dictionary translators for the description fields in source and target
        - Compare the code value and description for each pair of source and target and for each exact (or approximate) match, auto-populate the translation tables with the corresponding values
        - Insert all reference values from the source for which no target value could be identified, leaving the target system fields blank
        - Insert all reference values from the target for which no target value could be identified, leaving the source system fields blank

In our opinion, at present, complete population of translation tables is not possible due to the inherent limitations of the metadata and description fields. Consequently, data stewards would need to complete the definition of the translation tables in a semi-automatic manner, once step 5d is complete.

ANALYSIS AND DISCUSSION

In this section, we present a qualitative as well as quantitative analysis of our approach to the existing approaches.

Qualitative Comparison

The solution presented in this paper has various qualitative improvements compared to existing manual approaches to the semantic alignment problem. Firstly, due to the automated pre-processing step, the list of lookup table candidates is identified automatically, rather than the usual manual approaches employed today. This is a significant improvement over having to do this manually (or not doing it at all). Secondly, with our approach, the lookup table candidates undergo a domain identification process by validating against a domain ontology. This substantially improves the understanding of the source and target data models since it identifies whether certain lookup table encodes country codes, or units of measure, etc. From a data quality perspective, performing the semantic alignment has significant benefits. Applying duplicate detection to records coming out of different sources produces substantially better results if the data is standardized before duplicate detection is applied. The semantic alignment step is very helpful here since it improves the results of cleansing (e.g., name and address standardization), which is a typical standardization task executed before duplicate detection on customer master data is performed. Without having consistent country codes (typical example of a lookup table), it is challenging to use the right standardization rule set for country-specific address standardization. When the duplicate detection is running, the match results are better if the records coming from different sources have the same underlying semantics in all the fields. Finally, automating the semantic alignment problem to the extent possible saves costs and several weeks to months of manual work.

Quantitative Analysis

Assuming there are n candidate reference data tables each in source and target systems, each table has size m $(< n)$ and there is an ontology comprising of b nodes $(b >= n > m)$ and w edges, computation of the sets of value partitions ($\{Q_i\}$ in procedure P_1) is a one-time task, which can be computed in linear-time in terms of nodes b and edges w, $O(b + w)$. Next, computation of maximal value partitions for each reference data column would take pair-wise comparisons between the set of value partitions, $\{Q_i\}$ and the m data column values for each reference data table. This will take $b * m$ comparisons each for source and target tables. Thus, this step has a time complexity bound of $O(bmn)$. However, in practice, this search space can be reduced significantly by keeping track of the value partition nodes (much less than b) and using only them for comparisons. Once these domain abstractions have been computed, determining a target match for each source table is done in procedure P_2 by performing a simple BFS (again, linear in $b + w$) on the ontology graph. Thus, with reasonable complexity, our approach encompasses a range of qualitative benefits (described above) and relies on a discovery-based theme, exploiting domain ontology graph to deduce implicit semantic associations.

CONCLUSION

In this paper, we introduced an ontology-guided reference data alignment framework, solving a difficult information integration sub-task in information integration projects. We discussed approaches to enable the detection of reference data tables and their relative alignment across source/target systems, and gave an analysis to demonstrate the value of our approach. We also discussed that the final population of the translation tables is currently only semi-automatic, due to the limitations on the metadata quality, and thus, our approach establishes a best-effort solution to the problem.

ACKNOWLEDGEMENT

We would like to acknowledge contributions from Dan Wolfson and Dan Mandelstein, for their insights and comments during the development of this paper.

REFERENCES

1. Dreibelbis, A., Hechler, E., Milman, I., Oberhofer, M., van Run, P. and Wolfson, D. (2008) Enterprise Master Data Management: An SOA Approach to Managing Core Information, 1st Edition, Pearson.
2. Chisholm, M. (2000) Managing Reference Data in Enterprise Databases, Morgan Kaufmann, first edition, ISBN-10: 155860697.
3. Fryman, L., Inmon, W.H. and O'Neil, B. (2007) Business Metadata: Capturing Enterprise Knowledge, 1st Edition, Morgan Kaufmann.
4. Godinez, M., Hechler, E., Koenig, K., Lockwood, S., Oberhofer, M. and Schroeck, M. (2010) The Art of Enterprise Information Architecture: A Systems-Based Approach for Unlocking Business Insight, 1st Edition, Pearson.
5. Leser, U., Naumann, F.(2007) Informationsintegration. Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen, first Edition, dpunkt.Verlag.

6. Bauckmann, J., Leser, U. and Naumann F. (2010) Efficient and exact computation of inclusion dependencies for data integration, Number 34 in Technische berichte des Hasso-Plattner-Instituts fur Softwaresystemtechnik an der Universitat Potsdam.
7. Borek A., Woodall P., Oberhofer M. and Parlikad A. K. (2011) A Classification of Data Quality Assessment Methods. *Proceedings of the Sixteenth International Conference on Information Quality (ICIQ 2011)*, Adelaide, Australia.
8. Simperl, E., Mochol, M., and Burger, T. (2010) Achieving Maturity: the State of Practice in Ontology Engineering in 2009, *In International Journal of Computer Science and Applications*, 7(1), 45-65.
9. Ontology Engineering and Patterns Task Force. (2005) Representing Specified Values in OWL: "value partitions" and "value sets", W3C Working Group Note.
10. Web Ontology Working Group. (2004) OWL Web Ontology Language Semantics and Abstract Syntax, Section 4., Mapping to RDF Graphs, W3C Recommendation.
11. Oberhofer, M., Narayanan R., Pandit, S. (2012) Metadata exploitation in large-scale data migration projects, *Manuscript submitted to Americas Conference on Information Systems*, Seattle, Washington.
12. Pandit, S. and Honavar, V. (2010) Ontology guided extraction of complex nested relationships, *Proceedings of the 22nd International Conference on Tools with Artificial Intelligence*, Arras, France, IEEE Computer Society, 173-178.
13. Fot, D., Milman, I., Oberhofer, M., Pandit, S., Mandelstein, D. (2011) Ontology guided reference data discovery, United States Patent Application Publication, 20110295866.
14. Berson, A., Dubov L. (2010) Master Data Management and Customer Data Integration for a Global Enterprise, 2nd Edition, McGraw-Hill.
15. Li, Y., Liu, D., Zhang, W. (2003) A Generic Algorithm for Heterogeneous Schema Matching, *International Journal of Information Technology,* Vol. 9, No. 1.
16. Rahm, E. and Bernstein, P. (2001) A survey of approaches to automatic schema matching, VLDB Journal, 10: 334–350.
17. Bellahsene, Z., Bonifati, A., Rahm, E. (2011) Schema Matching and Mapping (Data-Centric Systems and Applications), ISBN 978-3-642-16517-7.