On the Quality and Effectiveness of Data Transformation Systems

Giansalvatore Mecca [1], Paolo Papotti [2],
Salvatore Raunich [3], and Donatello Santoro [1,4]

[1] Università della Basilicata – Potenza, Italy
[2] Qatar Computing Research Institute (QCRI) – Doha, Qatar
[3] Data Virtuality GmbH – Leipzig, Germany
[4] Università Roma Tre – Roma, Italy

(Discussion Paper)

1 Introduction

The problem of translating data among heterogeneous representations is a long standing issue in the IT industry and in database research. The first data translation systems date back to the seventies. In these years, many different proposals have emerged to alleviate the burden of manually expressing complex transformations among different repositories, so that today we have a very broad class of systems, ranging from schema-matching to schema-mappings and data-exchange, from data-integration to ETL, from object-relational mapping to data-fusion, and data-cleaning.

These systems differ under many perspectives. There are very procedural and very expressive systems, like those used in ETL. There are more declarative, but somehow less expressive schema-mapping systems. Some commercial systems are essentially graphical user interfaces for defining XSLT queries. Others, like data-exchange systems, incorporate sophisticated algorithms to enforce constraints and generate solutions of optimal quality. Some systems are inherently relational, some use nested data-models to handle XML data, and in some cases even ontologies. In light of this, database researchers have expressed a strong need to define a unifying framework for data translation and integration applications [4].

Since the range of application scenarios is very broad, and systems differ in terms of data models, expressive power, and efficiency, it would be very useful, given a task that requires to translate some input instance of a *source* schema into an output instance of the *target* schema, to have a common model to answer the following fundamental question: "*what is the right tool for my translation task ?*".

Answering this question entails being able to compare and classify systems coming from different inspirations and different application domains. In this paper, we concentrate on an ambitious task that has not been addressed so far, i.e., we aim at measuring the *level of intelligence* of a data transformation system. In our vision, the level of intelligence of the internal algorithms of a tool can be roughly defined as the ratio between the *quality* of the outputs generated by the system, and the *amount of user effort* required to generate them. In other terms, we want to measure, for each system, how much effort it takes to obtain results of the highest possible quality.

Notice that there exist some early benchmarks, designed both for ETL tools [12] and schema-mapping systems [1], that provide a basis for evaluating systems in the respective areas. Their main limitation, however, is that they fail in answering the main question addressed in this paper. They mainly concentrate on expressibility, by introducing representative, small scenarios [1].

To make our definition of quality and effort precise, we need several crucial notions: (i) a definition of *data-transformation* system, in such a way that this is sufficiently general to capture a wide variety of the tools under exam, and at the same time tight enough for the purpose of our evaluation; (ii) a definition of the *quality* of a data translation tool on a mapping scenario; (iii) a definition of the *user-effort* needed to achieve such quality.

In this paper, we report on our recent results [9] that contribute to giving an answer to the question above. More specifically:

(a) we introduce a very general definition of a *data-transformation* system, in terms of its input-output behavior; differently from earlier approaches, that have concentrated their attention on the actual specification of the transformations, we see a system as a black-box receiving as input some specification of the mapping task, and an instance of the source schema, and producing as output an instance of the target schema;

(b) we define the notion of *quality* of a data transformation tool on a given scenario as the *similarity* of the output instance wrt the *expected instance*, i.e., the "right" solution that the human developer expects for a given input. Notice that we allow nested data models and XML data, and therefore measuring the quality of an output imposes to compare two different trees, a known difficult problem, for which high-complexity techniques are usually needed. We show, however, that for the purpose of this evaluation it is possible to define an elegant and very efficient set-theoretic similarity measure that provides very accurate evaluations. Such comparison technique is a much needed contribution in this field, since it is orders of magnitude faster than typical edit-distance measures, and scales up to large instances;

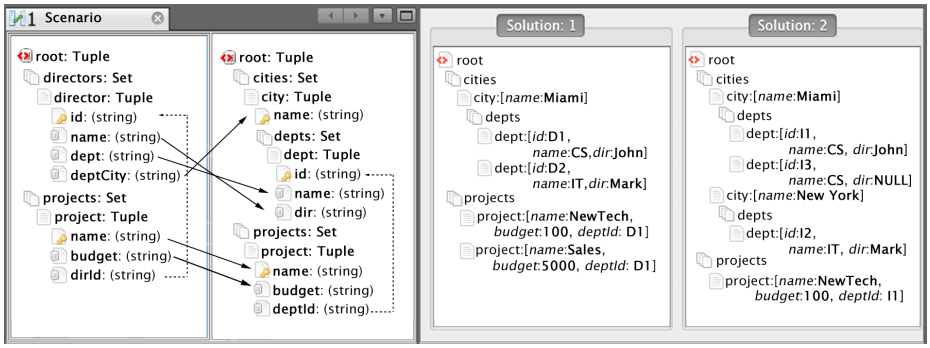

Fig. 1. Sample Scenario in the GUI of an Open-Source Mapping System.

(c) transformation systems typically require users to provide an abstract specification of the mapping, usually through some graphical user interface; to see an example, consider Figure 1, that shows the graphical specification of a sample mapping scenario.

Despite the fact that different tools have usually different primitives to specify the transformations, it is still possible to abstract the mapping specification as a *labeled input graph*; our estimate of the user effort is a measure of the size of an encoding of this graph inspired by the minimum description length principle in information theory [5]. We believe that this measure approximates the level of user effort better than previous measures that were based on point-and-click counts [1].

Based on these ideas, we are able to effectively classify data transformation tools that exist on the market and in the research community in terms of their *quality-effort* graphs. More specifically, we fix a number of representative scenarios, with different levels of complexity, and different challenges in terms of expressive power. Then, we run the various tools on different source instances, and with specifications of increasing efforts. Then, we measure the quality of the outputs.

This process allows us to derive several evidences. First, we know if the system is expressive enough to perform the translation, i.e., if it can at some point achieve 100% quality. Second, we measure how much intelligence the internal algorithms put into the solution, i.e., how fast is the increase in quality with respect to the increasing effort.

We believe that this kind of evaluation provides precious insights in the vast and heterogeneous world of transformation systems.

2 Overview

In this section we introduce the three main building blocks of our approach, namely: the definition of a data transformation system, the quality measure, and the user-effort measure. For further details we refer the reader to [9].

2.1 Transformation Systems and Scenarios

In our view, a *data-transformation system* is any tool capable of executing *transformation scenarios* (also called mapping scenarios or translation tasks). Regardless of the way in which transformations are expressed, in our setting transformation scenarios require to translate instances of a source schema into instances of a target schema. The transformation system is seen as a black box, of which we are only interested in the input-output behavior. For each transformation task, inputs to the system are: (a) the source schema, S; (b) the target schema, T; (c) an input instance, I; (d) a specification of the mapping, M. The output is the target instance, J, generated by the system by applying M to I. Notice that the source and target schema may be either explicit, or implicit, as it happens in many ETL tasks. As it is common in mapping tools [10, 1], we assume a nested-relational data model for S and T, capable of handling both relational data and XML trees. Notice how our definition of the mapping, M, is very general. In fact, we are not constraining the way in which a tool allows users to express the needed transformations. This can be done by providing some abstract specification through a GUI, or – going to the opposite extreme – even by manually writing a query in SQL or XQuery.

As it was discussed in the previous section, our quality measure requires, for each translation task, to identify the expected solution, i.e., the "gold standard" to which the actual outputs must be compared. It is possible to assume that such expected solution

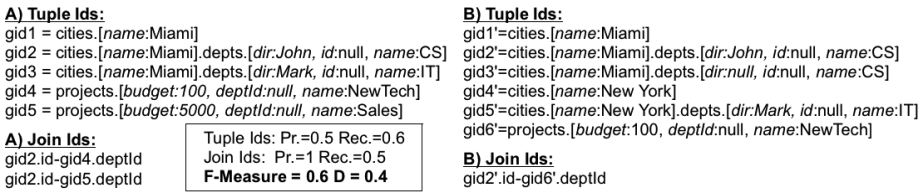

Fig. 2. Comparing Instances

can be selected by a human designer. Further details about how to generate this automatically are provided in [9].

2.2 The Quality Measure

Our idea is to evaluate the quality of a tool by measuring the similarity of its outputs wrt a fixed, expected instance. Given the nested relational data model, our instances are trees. Figure 1 shows an example. While there are many existing similarity measures for trees, it is important to emphasize that none of these can be used in this framework, for the following reasons:

(i) we want to perform frequent and repeated evaluations of each tool, for each selected scenario and mapping specifications of different complexity. Also, we want to be able to work with possibly large instances, to measure how efficient is the transformation generated by a system. As a consequence, we cannot rely on known tree-similarity measures, like, for example, edit distances, that are typically quite expensive;

(ii) the problem above is even more serious, if we think that our instances may be seen as graphs, rather than trees; we need in fact to check key-foreign key references, that can be seen as additional edges, thus making each instance a fully-fledged graph; graph edit distance is knowingly much more complex than tree edit distance;

(iii) even though we were able to circumvent the complexity issues, typical tree and graph-comparison techniques would not work in this setting. To see this, consider that it is typical in data-transformation tasks to generate synthetic values in the output – these values are called *labeled nulls* in data-exchange and *surrogate keys* in ETL. In Figure 1, values $D1, D2, I1, I2, I3$ are of this kind. These values are essentially placeholders used to join tuples, and their actual values do not have any business meaning. We therefore need to check if two instances are identical up to the renaming of their synthetic values. We may say that we are rather looking for a technique to check *tree or graph isomorphisms*, an even more complex problem.

It can be seen that we face a very challenging task: devising a new similarity measure for trees that is efficient, and at the same time precise enough for the purpose of our evaluation. In order to do this, we introduce the following key-idea: since the instances that we want to compare are not arbitrary trees, but rather the result of a transformation, they are supposed to exhibit a number of regularities; as an example, they are supposedly instances of a fixed nested schema, that we know in advance. This means that we know: (a) how tuples in the instances must be structured; (b) how they should be nested into one another; (c) in which ways they join via key-foreign key relationships.

We design our similarity metric by abstracting these features of the two trees in a set-oriented fashion, and then compare these features using precision, recall and ultimately

F-measures to derive the overall similarity. More specifically, for each instance, we compute: (i) first a set of tuple identifiers, also called *local identifiers*, one for each tuple in the instance; (ii) a set of nested tuple identifiers, called *global identifiers*, that capture the nesting relationships among tuples; (iii) a set of pairs of tuple identifiers, called *join pairs*, one for each tuple t_1 that joins a tuple t_2 via a foreign key. Tuple identifiers and join pairs for our example are reported in Figure 2.

It is worth noting that the computation of tuple identifiers requires special care. As it can be seen in the Figure, we keep these values out of our identifiers, in such a way that two instances are considered to be identical provided that they generate the same tuples and the same join pairs, regardless of the actual synthetic values generated by the system.

Based on these ideas, to simplify the treatment we may say that, both for the generated instance and for the expected one, we generate two sets: the set of global identifiers, and the set of join pairs. Then, we compare the respective sets to compute precision and recall, and compute an overall F-Measure that gives us the level of similarity. Figure 2 reports the values of precision and recall and the overall F-measure for our example.

2.3 Estimating User Efforts

To measure the complexity of the mapping specification, and thus estimate user efforts, we introduce an information-theoretic technique. To do this, we model the specification complexity as an *input-graph* with labeled nodes and labeled edges. Experience shows that this model is general enough to cover every data transformation, spanning from schema mapping transformations to ETL ones.

Every element in the source and target schemas is a node in the graph. Arrows among elements in the GUI become edges among nodes in the graph. The tool may provide a library of graphical elements – for example to introduce system functions – that are modeled as additional nodes in the graph. Extra information entered by the user (e.g., manually typed text) is represented as labels over nodes and edges.

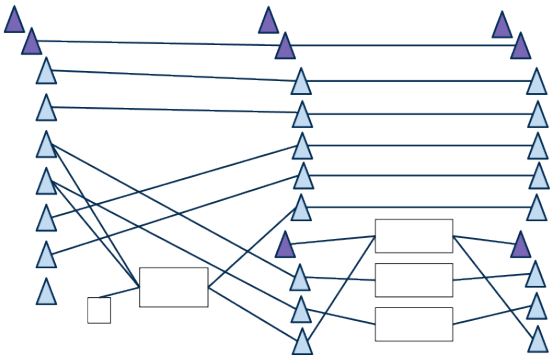

Fig. 3. Input graph for the vertical partition scenario in a commercial mapping system.

We measure the size of such graphs by encoding their elements according to a minimum description length technique [5], and then by measuring the size in bits of such description.

We report in Figure 3 the input-graph for a sample scenario in Figure 1 (this is the vertical partition scenario from [1], expressed using a commercial mapping system). The graph consists of the following elements: (i) 31 nodes for the source and target schemas; note that the schemas are considered as part of the input and are not counted in the encoding; (ii) there are 5 functions in this scenario that generate additional nodes; this makes a total of 36 nodes, so that we shall

need 6 bits for their encoding; function nodes are labeled by the associated function; based on the size of the library of functions provided by the tool, every function node requires an additional 8 bits for its encoding; (iii) 26 edges (without labels), that we shall encode as pairs of node identifiers (2*6 bits); (iv) finally, to specify the mapping, one of the schema nodes must be labeled by a char, which we need to encode.

The specification complexity is therefore given by the following sum of costs: $(5 * (6 + 8)) + (26 * (2 * 6)) + (6 + 8) = 396$ bits. With the same technique we are able to measure the size of the specification needed by different systems, and compare efforts.

3 Experimental Results

The techniques presented in the paper have been implemented in a working prototype using Java; the prototype has been used to perform a large experimental evaluation. To start, we show how the proposed quality measure scales well up to very large instances and outperforms existing techniques. Then, we use our framework to compare several data-translation systems on a common set of scenarios and discuss the results. All the experiments have been conducted on a Intel Xeon machine with four 2.66Ghz cores and 4 GB of RAM under Linux.

In the spirit of evaluating representative systems from different perspectives, we have included the following tools: (i) an open-source schema-mapping research prototype [6, 7]; (ii) a commercial schema-mapping system; (iii) a commercial ETL tool.

In addition, to discuss the relationship of our technique to STBenchmark, we have also evaluated the performances of a different schema-mapping tool [10] on some of the scenarios. It is worth mentioning that we also considered the idea of including in our evaluation the open-source OpenII data integration tool [11]. We found out that, while promising, the data translation module of the current version of OpenII is still in a rather preliminary state, and therefore decided to exclude it from the evaluation.

To conduct the evaluation, we selected twelve transformation tasks from the literature, with different levels of complexity. The selected tasks can be roughly classified in three categories:

(i) basic mapping-operations, taken from STBenchmark [1]: *Copy* (Cp), *Denormalization* (Dn), *Vertical Partition* (VP), *Key Assignment* (KA), *Value Management* (VM), and *Nesting* (Ne);

(ii) advanced mapping operations, taken from [8, 6]: *Minimization* (Mi), *Fusion* (Fu), *Nested Keys* (NK); these scenarios require the management of more complex target constraints wrt those above;

(iii) typical ETL tasks: *Aggregation* (Ag) is a simplified version of the *line* workflow from an ETL benchmark [12], while *AccessLog* (AL) and *CreditCards* (CC) are taken from the http://cloveretl.com/examples Web page.

Of these twelve scenarios, five required the introduction of surrogates; two use nested models, the others are relational. For each scenario we have identified the gold standard, that is, the desired target solution for the given translation task and source instance. Expected solutions were identified in such a way that they contained no unsound

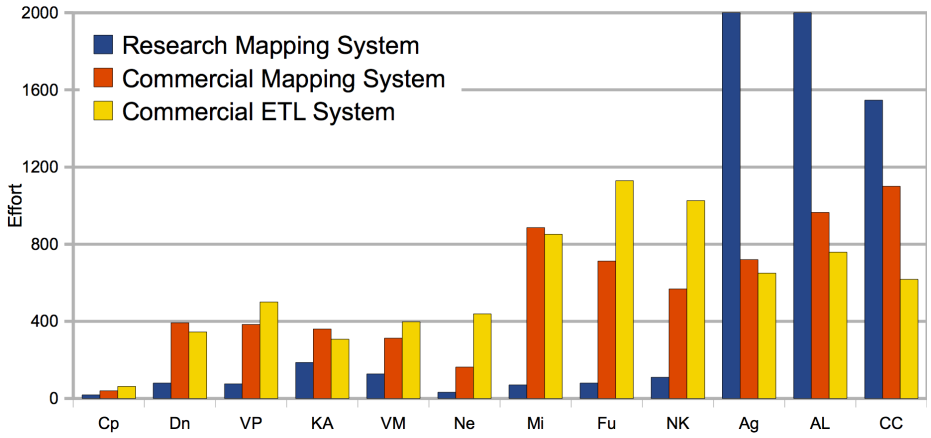

Fig. 4. Effort needed by the systems to obtain 100% quality in the various scenarios.

or redundant information [3]. Then, we tried to generate the expected output with each of the tools, and measured efforts and quality.

Notice that the three tools under exam use fairly different strategies to compute the transformations: the research mapping systems generate SQL or XQuery code, the commercial mapping system typically generates XSLT, while the ETL workflow requires the internal engine in order to produce results. Nevertheless, our general view of the transformation process permits their comparison. We also want to mention that, while it is not in the scope of this paper to compare the systems in terms of scalability, all of the systems in our experiments scaled well to large instances.

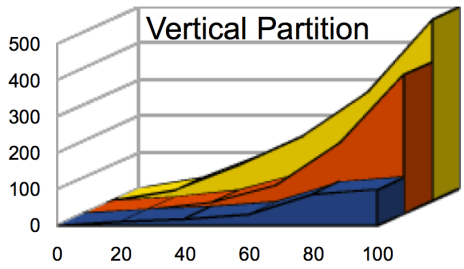

Fig. 5. Sample Quality-Effort Graph

Results are shown in Figure 4. We report the effort needed by the various systems to obtain 100% quality in the various scenarios. As a first evidence, we note that the research mapping tool required considerably less effort than the commercial counterparts on the basic and advanced mapping tasks. On these tasks, the ETL tool was the one requiring the highest effort to compute the requested transformations. However, we also note that the situation is reversed for the ETL-oriented tasks. Notice how the commercial mapping system had intermediate performances. This suggests that these tools are progressively evolving from the schema-mapping ecosphere into fully-fledged ETL tools.

To get more insights about this, let us look at the quality-effort graph in Figure 5. The Figure shows how much effort is needed to get a certain level of quality with a given system. Recall that one of our aims is that of measuring the “level of intelligence” of a tool, as its quality/effort ratio. From the graphical viewpoint, this notion of IQ can be associated with the area of the graph delimited by the effort-quality function: such area can be taken as a measure of the progressive effort needed by a tool to achieve

increasing levels of quality in one experiment. The smaller the area, the higher is the IQ of a system.

Further experimental results are provided in [9]. Our experiments confirm the intuition that the sophisticated declarative algorithms introduced in recent years in schema-mappings research may really provide some advantage in terms of productivity to the data architect. However, this advantage is somehow confined to the typical scope of applicability of schema-mappings. When users want to deal with more complex scenarios, i.e., transformations requiring a rather fine-grained manipulation of values, the adoption of more procedural paradigms brings some advantages.

We strongly believe that these results advocate the need for a new strategy for developing data-transformation tools, which brings together the best of both worlds, in the spirit of [2]. While the expressive power of procedural ETL tools is necessary to properly handle the wide range of transformations that a data architect typically faces, still there are a variety of mapping tasks – ranging from conjunctive queries, data-fusion and instance minimization, to management of functional dependencies and nested constraints – for which research mapping tools provide building blocks that may represent a very powerful addition to commercial transformation systems.

References

1. B. Alexe, W. Tan, and Y. Velegrakis. STBenchmark: Towards a Benchmark for Mapping Systems. *PVLDB*, 1(1):230–244, 2008.
2. S. Dessloch, M. A. Hernandez, R. Wisnesky, A. Radwan, and J. Zhou. Orchid: Integrating Schema Mapping and ETL. In *ICDE*, pages 1307–1316, 2008.
3. R. Fagin, P. Kolaitis, and L. Popa. Data Exchange: Getting to the Core. *ACM TODS*, 30(1):174–210, 2005.
4. L. M. Haas. Beauty and the Beast: The Theory and Practice of Information Integration. In *ICDT*, pages 28–43, 2007.
5. D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
6. B. Marnette, G. Mecca, and P. Papotti. Scalable data exchange with functional dependencies. *Proceedings of the VLDB Endowment*, 3(1):105–116, 2010.
7. B. Marnette, G. Mecca, P. Papotti, S. Raunich, and D. Santoro. ++SPICY: an opensource tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment*, 4(11):1438–1441, 2011.
8. G. Mecca, P. Papotti, and S. Raunich. Core Schema Mappings. In *ACM-SIGMOD International Conference on the Mangement of Data (SIGMOD 2009)*, pages 655–668, 2009.
9. G. Mecca, P. Papotti, S. Raunich, and S. Santoro. What is the IQ of your Data Transformation System? In *International Conference on Information and Knowledge Management (CIKM 2012)*, pages 872–881, 2012.
10. L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.
11. L. Seligman, P. Mork, A. Halevy, K. Smith, M. J. Carey, K. Chen, C. Wolf, J. Madhavan, A. Kannan, and D. Burdick. OpenII: an Open Source Information Integration Toolkit. In *SIGMOD*, pages 1057–1060, 2010.
12. A. Simitsis, P. Vassiliadis, U. Dayal, A. Karagiannis, and V. Tziovara. Benchmarking etl workflows. In *TPCTC*, pages 199–220, 2009.