Position Statement: Efficient development of data migration transformations

Paulo Carreira
Oblog Consulting and FCUL
paulo.carreira@oblog.pt

Helena Galhardas
INESC-ID and IST
hig@inesc-id.pt

1 Introduction

Data migration and integration projects typically involve two phases. The first phase aims at establishing the schema and data mappings required for transforming schema and data. The second phase consists of developing and executing the corresponding schema and data transformations.

Several tools have been designed to assist the discovery of appropriate schema mappings [RB01], while data understanding solutions (e.g., Integrity [Val]) have been progressively adopted for helping to find out the correct data mappings. The development of the corresponding schema and data transformations is usually an ad-hoc process that comprises the construction of complex programs and queries.

Ideally, a framework should exist to assist the development, execution and correction of schema and data transformations. The execution of a given data or schema transformation usually gives further insight about the problem domain. For example, erroneous mappings are frequently detected when transformations are executed. In this case, the programs that implement the corresponding schema or data transformations must be modified. In real-world scenarios, this approach turns out to be infeasible due to the large number of modifications that must be introduced.

2 How we position ourselves

The computer-assisted development, execution and correction of schema and data mappings lay in the iterative refinement of mappings until the appropriate set of executable schema and data transformations is obtained.

Our work has been concerned with a particular case of the schema and data integration problem that consists of transforming a source schema into a fixed target schema. This problem frequently arises when a legacy system is migrated

into a modern system. We have developed a high-level language for specifying and refining schema and data mappings, and a system that supports its efficient execution. Since we have been involved in real-world projects, functionalities like a productive IDE (Integrated Development Environment) and mechanisms for project tracking and auditing have also received particular attention.

An initial version of our data migration tool-box named Data Fusion is already implemented and currently used in industrial settings. However, only a subset of the specification language is supported. The tool-box research and development are sponsored by Oblog Consulting.

3 Challenging issues

Due to their inherent complexity, schema and data mappings must be iteratively specified and refined. Shortening the time needed for each iteration results in a global reduction of the effort required to develop an entire data migration or integration project. The following features are required:

an adequate mapping specification language that provides the abstractions for conveniently specifying an refining the solutions to common data integration and transformation problems. This high-level language is also expected to fulfill the gap between business and implementation experts, thus reducing communication costs. Moreover, the language must be powerful enough so that ad-hoc programs are not needed. Domain specific languages [vDKV00] seem to be the approach to follow.

the partial execution of schema and data mappings enables the efficient deployment of potentially large and complex mappings and avoids the cost of entire compilations. This functionality is useful, for example, when predicting the effect of transforming data for a subset of fields. We plan to adapt and integrate a technique known as *program slicing* [Wei82, Tip95, Luc01] into our specification language for automatically computing simpler mappings given specific criteria.

the efficient execution of mappings is a major requirement in real-world scenarios for integrating and transforming millions of records in a limited time-frame. Several optimizations can be introduced both at compile and run time.

a debugging facility can greatly reduce the cost of locating anomalies in complex schema and data transformations. It should include a debugger and support partial executions and lineage tracing features [CW01].

4 About the authors

Paulo Carreira is a senior engineer at Oblog Consulting and a PhD student at FCUL (Faculty of Sciences of the University of Lisbon). He got his MSc on

automatic verification of object-oriented specifications. Helena Galhardas is a researcher at INESC-ID and professor at IST (Instituto Superior Técnico), the engineering school of the Technical University of Lisbon. Helena got her PhD on data cleaning.

References

[CW01] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001.

[Luc01] A. De Lucia. Program slicing: Methods and applications. In *In 1st IEEE Int'l Workshop on Source Code Analysis and Manipulation*, pages 142–149, Florence, Italy, 2001. IEEE Computer Society Press, Los Alamitos, California, USA.

[RB01] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[Tip95] F. Tip. A survey of program slicing techniques. *Journal of programming languages*, 3:121–189, 1995.

[Val] Vality. Home page of the integrity tool. http://www.vality.com/html/prodint.html.

[vDKV00] Arie van Deursen, Paul Klint, and Joost Visser. Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, 35(6):26–36, 2000.

[Wei82] M. Weiser. Programmers use Slicing when debugging. *Comunications of the ACM*, 25(7):446–452, July 1982.