

# Tools and Techniques for *Understanding* Data Exchange Systems

Wang-Chiew Tan  
University of California, Santa Cruz  
Email: wctan@cs.ucsc.edu

## Abstract

It is common knowledge that distinct but often content-related data sources exist in different formats. Consequently, the ability to restructure (or exchange) data from these sources into a format desired by consumers of such data sources is extremely valuable. It is also common knowledge that a lot of time and effort are needed to set up and tune data exchange and data integration systems. Therefore, it is important to devise techniques and tools that can reduce the amount of effort needed to set up, and subsequently, understand and maintain such systems. Our position is that more tools and techniques are needed to facilitate the installation, understanding and maintenance of data exchange systems.

## 1 Data Exchange

The problem of restructuring data from one format into another is a long standing one. To date, most research on this topic are geared towards techniques for providing a unified view of different data sources, finding relationships between data sources, different approaches for describing views of data sources and answering queries posed against such views, as well as reasoning about queries in such systems. In addition to techniques for resolving the issues described earlier, techniques and tools for understanding various components of a data restructuring (also called data exchange) system are also needed to (1) reduce the amount of effort needed to install a data exchange system and, (2) understand and maintain a data exchange system.

A fundamental component in current research on data exchange is that of *schema mappings* [5]. A schema mapping is a high-level declarative specification of the relationship between two schemas. It specifies which selected data residing under the *source schema* is to be extracted and how they are to be restructured to fit under the *target schema*. A recent example of a data exchange system that allows a user to program schema mappings in a language that is based on a widely adopted formalism in literature is Clio [4]. Schema mappings are desirable for specifying an exchange because they have a rigorous semantics that can be useful for understanding the properties and behavior of a data exchange system, or even multiple data exchange systems that are connected together. Schema mappings are also desirable, as opposed to lower-level languages such as XSLT scripts, because they are declarative and they provide an abstraction for the underlying implementation of the exchange. In short, the clean abstraction provided by schema mappings tends to reduce the amount of effort needed to install, understand and maintain a data exchange system.

Although schema mappings form the backbone of data exchange and data integration systems, tools for managing schema mappings are still lacking. In particular, more tools and techniques need to be developed to help users of such systems *understand* schema mappings, and more generally, data exchange systems. For example, a recent effort towards this goal is SPIDER [1, 3], a debugging tool for schema mappings. SPIDER provides an interface with standard debugging features such as breakpoints and watch, much like a debugging tool for programming languages. A unique feature of SPIDER is the ability to describe the

relationship (the provenance and flow of data) between restructured data and data sources, through schema mappings. A user understands a schema mapping by “executing” the exchange over sample or real data sources and analyzing the behavior of the execution. This is analogous to how a standard debugger allows a user to understand programs through analyzing its behavior in various test cases. SPIDER can be easily deployed in any data exchange or data integration system that is based on schema mappings.

## 2 Challenges

Our position is that more tools and techniques are needed to facilitate the installation, understanding and maintenance of data exchange systems.

For installing a data exchange system, automation on techniques for discovering the “right” relationship between schemas and choosing the “right” interpretation of the relationship for the exchange is needed. There has been research on the former (schema matching) and less research on the latter. For understanding and maintaining a data exchange system, we list a few challenges here, which we have arrived by way of understanding how a programmer debugs a program, as well as from our experience with developing SPIDER. We believe that a *data-driven* approach is useful for understanding schema mappings. Currently, SPIDER is able to help a user understand a schema mapping by illustrating the schema mapping through the relationship between source data and restructured data. It would be useful if one could also reflect the changes that would be applied on the source data and restructured data when a schema mapping is modified. Such scenario happens often during a debugging session, where the schema mapping may be modified. Understanding a schema mapping can also be achieved through “cause-and effect”. With a fixed schema mapping, one may change the contents of the data sources, or even the restructured data, during debugging so that the cause-and-effect of their changes can help them understand the schema mapping. It would be useful if one could reflect the changes that would be applied to the data sources or the restructured data when such changes happen. We emphasize that these problems arise not only in the context of debugging, even though we have motivated them through the debugging standpoint. In fact, these problems correspond to the classical *view adaptation*, *view maintenance* and *view update* problems in existing literature and should be re-examined in the context of the data exchange framework. Lastly, we also believe that more research on the various operators for schema mappings, which are similar to those defined for model management [2], are needed. The results from these studies will be crucial towards the understanding of the relationship and behavior of inter-related data exchange systems.

## References

- [1] B. Alexe, L. Chiticariu, and W. Tan. SPIDER: a Schema mapPIng DEbuggeR. In *VLDB–Demonstration Track–(To appear)*, 2006.
- [2] P. A. Bernstein. Applying model management to classical meta data problems. In *CIDR*, pages 209–220, 2003.
- [3] L. Chiticariu and W. Tan. Debugging Schema Mappings with Routes. In *VLDB (To appear)*, 2006.
- [4] L. M. Haas, M. A. Hernandez, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *SIGMOD*, pages 805–810, 2005.
- [5] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.
- [6] Y. Velegarakis, R. J. Miller, and J. Mylopoulos. Representing and querying data transformations. In *ICDE*, pages 81–92, 2005.