

EII and ETL Unification---It's about Time!

Howard Ho, ho@almaden.ibm.com

IBM Almaden Research Center

Joint with Mauricio Hernandez and Lucian Popa

Introduction

Integration, transformation and exchange of data are well known and fast growing problems. Solving the integration problem typically requires: (1) understanding and reconciling the differences between various schemas, and (2) generation of mappings between these schemas, which will then drive any subsequent processing of data. The combination of steps (1) and (2) is a labor-intensive and error-prone process.

Our research group has years of expertise building the Clio schema mapping system [1, 2] that semi-automatically guides users towards the creation and deployment of schema mappings. We investigated and implemented methods for generating transformation queries given a schema mapping (for both relational and XML schemas) and methods for discovering such mappings (schema matching). We also studied the foundational aspects that underlie schema mappings and the data exchange process that is associated with schema mappings.

During the last few years, we have continued to explore more advanced information integration tooling features using Clio as our research proof-of-concept platform. Examples of such features include advanced design components as well as runtime extensions: support for schema integration, handling schema evolution, support for mapping composition and inversion, mapping-based query rewrite, and transformation runtime engine (for XML-to-XML transformation, XML-to-RDB shredding and RDB-to-XML publishing).

Much of this research by us and by others is crucial for the advance of EII (Enterprise Information Integration) industry. However, in its current form, it only addresses the EII market, but not the bigger, more mature, market in ETL (Extract, Transform and Load). A number of database leading researchers in industry, such as Mike Carey and Laura Haas, have voiced the desire to apply EII research into ETL and to unify EII and ETL. In what follows, we will define and motivate this problem. We then argue that the timing is right and the underlying EII technologies are mature enough for us, the information integration research community as a whole, to tackle this big challenge.

Initial Research: Mapping Abstraction for ETL

The mapping formalism that lies underneath our current solutions (as well as many commercially available solutions) is simple but sufficiently expressive for the complexity of many integration problems. In particular, the mapping language investigated by us [2, 4] as well as by most academic research (e.g., GLAV mappings [3]), has declarative, logic-based features (similar to SQL), which are easier to reason with and facilitate analysis, deployment and maintenance (such as handling schema evolution [5]). However, more procedural, data-flow like features, capturing ETL (extract-transform-load) semantics, as well as recursive capabilities needed by many industrial XML Schema standards, are missing. In particular, the ETL developer community typically expresses (or “programs”) an ETL task, procedurally, as a data-flow graph. During an ETL run, data is extracted from the source tables or XML documents, flows through a number of intermediate nodes in a streaming manner, and reaches resulting tables or files as its final form.

During a particular ETL stage, multiple input streams may be joined or merged, while multiple output streams may be created by splitting or replicating an input stream.

Users typically create an ETL task by constructing a data-flow-like graph consisting of sources, operators and targets in a very procedural, bottom-up manner. Although this is a powerful paradigm, it also requires experienced programmers. It often takes months of development, and it is hard to subsequently analyze, verify, maintain and re-deploy. A more abstract model behind ETL is missing. Some ETL operators can be naturally mapped to relational algebra expressions (such as join). However, others may be procedural (such as unique id generation) or beyond the range of relational algebra (such as the pivot operation that promotes data to metadata). Furthermore, the concepts of merge and split need to be captured by the abstract model as well. Thus, we see a continuing need for the development of the “right” programming paradigm for mappings and transformations that facilitates and automates many of the design aspects of information integration systems. Our initial research, which we call *mapping abstraction for ETL*, is to derive such abstract model from ETL tasks and, yet, make such model be consistent (modulo some natural extension) to the existing schema mapping models in EII. With such abstraction, the ultimate goal of *ETL and EII unification* may be achieved through a layered architecture, where EII mappings and ETL tasks are the top and bottom layers, respectively, and the new abstract model is in the middle.

Our Position: ETL and EII Unification

Our goals for the proposed research are described as follows:

- **Round-tripping between ETL scripts and declarative mappings**
 - Generate ETL scripts from mappings.
 - Extract mapping information from ETL scripts.
 - Track and propagate mapping-related modifications
- **Optimization, analysis and maintenance of ETL scripts**
 - Remove redundant operations
 - Push computation to other runtimes
 - “Rewrite” ETL scripts
 - Handling schema evolution

We hope our research in defining a declarative *mapping abstraction for ETL* will be the starting point of a wider research effort towards easing the use of ETL middleware, improving “bad ETL scripts” performance, and unifying tooling support for EII and ETL. The unification of EII and ETL should yield the best of both worlds: performance and programming flexibility from the ETL side, plus all the advantages from having a declarative mapping abstraction (more automation, optimization, reuse, etc.). Finally, the further advance of debugging tools and data lineage research in EII [6] should be reflected in ETL as the next step.

Bibliography

- [1] Miller RJ, Haas LM and Hernández MA. Schema Mapping as Query Discovery. VLDB 2000.
- [2] Haas LM, Hernández MA, Ho H, Popa L and Roth M. Clio Grows Up: From Research Prototype to Industrial Tool. SIGMOD 2005
- [3] Lenzerini M: Data Integration: A Theoretical Perspective. PODS 2002
- [4] Fagin R, Kolaitis PG, Miller RJ, Popa L. Data Exchange: Query Answering and Semantics. *Theoretical Computer Science*, 336(1): 89-124 (2005).
- [5] Velegrakis Y, Miller RJ, Popa L, Preserving Mapping Consistency under Schema Changes. *The VLDB Journal*, 13(3): 274–293, Sep 2004.
- [6] Chiticariu L, Tan W-C. Debugging Schema Mappings with Routes. VLDB 2006.