Schema Mappings and Data Examples*

Balder ten Cate
UC Santa Cruz

Phokion G. Kolaitis
UC Santa Cruz and
IBM Research - Almaden

Wang-Chiew Tan
UC Santa Cruz

ABSTRACT

A fundamental task in data integration and data exchange is the design of *schema mappings*, that is, high-level declarative specifications of the relationship between two database schemas. Several research prototypes and commercial systems have been developed to facilitate schema-mapping design; a common characteristic of these systems is that they produce a schema mapping based on attribute correspondences across schemas solicited from the user via a visual interface. This methodology, however, suffers from certain shortcomings. In the past few years, a fundamentally different methodology to designing and understanding schema mappings has emerged. This new methodology is based on the systematic use of *data examples* to derive, illustrate, and refine schema mappings.

Example-driven schema-mapping design is currently an active area of research in which several different approaches towards using data examples in schema-mapping design have been explored. After a brief overview of the earlier methodology, this tutorial will provide a comprehensive overview of the different ways in which data examples can be used in schema-mapping design. In particular, it will cover the basic concepts, technical results, and prototype systems that have been developed in the past few years, as well as open problems and directions for further research in this area.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases – Data Translation; H.2.4 [**Database Management**]: Systems – Relational Databases

General Terms

Algorithms, Languages, Theory

Keywords

Schema mappings, data examples, data exchange, data integration

*All three authors are supported by NSF grant IIS-0905276. In addition, the first two authors are supported by NSF grant IIS-1217869.

1. INTRODUCTION AND OVERVIEW

Background A key task in data integration or data exchange is the design of a schema mapping between database schemas. A *schema mapping* is a high-level, declarative specification of the relationship between two database schemas, typically called the *source schema* and the *target schema*. Schema mappings form the backbone for specifying and carrying out data-interoperability tasks involving data across different sources [6, 14, 15].

Syntactically, schema mappings are specified in some schema-mapping language, such as, for example, the language of GLAV (Global-and-Local-As-View) constraints. Due to the complexity and scale of many real-world schemas, a schema mapping may consist of a large number of constraints. Developing the "right" schema mapping for a given pair of schemas can be a daunting task. Several mapping-design systems have been developed to facilitate the process of designing schema mappings. These systems include research prototypes, such as Clio [13], HePToX [9], Spicy++[1], as well as commercial systems, such as Microsoft's mapping composer [7], Altova Mapforce[2], and Stylus Studio[3].

Each of the aforementioned schema-mapping design systems adheres to the following general methodology. The two schemas for which a schema mapping is to be designed are simultaneously displayed, and a visual specification of correspondences between elements of the two schemas is solicited from the user (e.g., see Figure 1, where the source schema is displayed on the left, the target schema is displayed on the right, and the relationships between the elements of the schemas are indicated in the middle). Some of the correspondences between elements of the two schemas may also be inferred semi-automatically with the help of a schema-matching module. Once the specification consisting of the schemas and the correspondences has been completed, the system produces a schema mapping between the two schemas.

While these schema-mappings design systems help the user by automatically generating a schema mapping from a visual specification, the resulting schema mapping often needs to be manually tuned before it accurately represents what the user has in mind. This manual tuning process is a difficult task, as it involves understanding and modifying the schema mapping generated by the system. Part of the problem is the fact that a visual specification consisting of correspondence between the two schemas, such as in Figure 1, provides only partial information about the schema mappings. As a matter of fact, more than one schema mapping may be consistent with the same visual specification. What is worse, different schema-mapping design systems may generate different (logically inequivalent) schema mappings from the same visual spec-

[1] http://www.db.unibas.it/projects/spicy/
[2] http://www.altova.com/mapforce.html
[3] http://www.stylusstudio.com/xml_mapper.html

ification [2]. Therefore, in order to ensure that the semantics of the derived schema mapping is as intended, the user is required to manually inspect the generated schema mapping; this can be a time consuming and arduous task, in view of the length and the complexity of the schema mapping produced by the system.

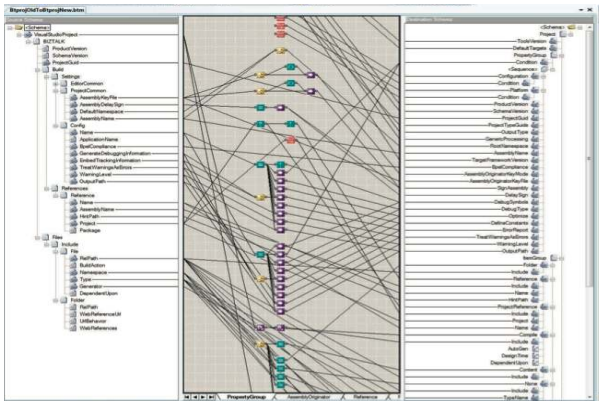

Figure 1: Visual specification of value correspondences. This figure is taken from [8].

Aim of the Tutorial The aim of this tutorial is to present a comprehensive overview of a fundamentally different methodology for the design and understanding of schema mappings. This methodology, which has been developed during the past few years, is based on the systematic use of *data examples* to facilitate the design and the understanding of schema mappings. Here, a *data example* is a pair consisting of a source instance and a target instance. Several different, yet inter-connected, recent strands of research have shown the usefulness of data examples in schema-mapping design. Data examples can be used to illustrate, refine, and derive schema mappings. Papers in which data examples are used to illustrate and refine schema mappings include [1, 17], while papers in which data examples are used to derive schema mappings include [16, 11, 12, 4]. In particular, [12] derives a schema mapping from a single data example using a cost model that takes into account several parameters, including the size of the syntactic description of the schema mapping and also how well the schema mapping "explains" the example. In general, given a set of data examples, there may exist zero, one, or more than one schema mapping that "fits" a given collection of data examples. In [4], an algorithm was presented that, given a finite set of data examples, decides whether or not there exists a GLAV schema mapping (i.e., a schema mapping specified by GLAV constraints) that "fits" these data examples. If such a fitting GLAV schema mapping exists, then the system returns the "most general" such schema mapping. A prototype system based on these ideas was demonstrated at [5]. A different, but related, investigation has focused on the problem of determining which schema mappings can be uniquely characterized via a finite set of data examples [3]. More recently, in [10], the problem of obtaining a schema mapping from data examples is cast as an algorithmic learning problem, and is studied using concepts and methods of computational learning theory.

Example-driven design of schema mappings is becoming an important and active area of research. As the preceding discussion shows, a variety of approaches toward the use of data examples in schema-mapping design have been investigated. The tutorial will provide a comprehensive overview of the following topics:

(i) The different ways in which data examples can be used to facilitate schema-mapping design,

(ii) The various approaches, technical results, and prototype systems that have been developed over the past few years.

(iii) Open problems and directions for further research in this area.

This is a three-hour tutorial intended for the typical ICDT and EDBT participant who has an interest in information integration and data interoperability problems. To the extent possible, the tutorial will be self-contained for the general DB research audience. In particular, no background knowledge other than an understanding of the basics of database theory and systems will be assumed.

2. CONTENTS OF THE TUTORIAL

The tutorial has three main parts. In the first part of the tutorial, we will first give a gentle introduction to schema mappings, schema-mapping languages, and data exchange, focusing on aspects that are relevant for understanding the remaining parts of this tutorial. After this, we will describe the prior state-of-the-art methodology and systems for schema-mapping design. In the second part of the tutorial, we will give an account of the dual use of data examples: how data examples can be used to illustrate or to "capture" a given schema mapping, and, conversely, how a schema mapping can be derived from data examples. In the third and final part of the tutorial, we will discuss the interactive design of schema mappings via data examples from several different perspectives, including the lens of computational learning theory. Open problems will be discussed in each of the three parts of the tutorial.

2.1 Background and Motivation

Schema Mappings and Data Exchange This part of the tutorial will highlight the fundamental role of *schema mappings* in data inter-operability tasks, such as data exchange and data integration, and how they are used in practice. Data exchange is the problem of transforming data residing in a schema, called the source schema, into data structured under a different schema, called the target schema; in particular, data exchange entails the materialization of data, after the data have been extracted from the source format and re-structured into the target format. Data integration is a closely related but different problem. It can be described as symbolic or virtual integration: users are provided with the capability to pose queries and obtain answers via the unified format interface, while the data remain in the sources and no materialization of restructured data takes place. These problems are ubiquitous in several different contexts (from enterprise data to data gathered in scientific applications) and constitute a challenge for all large organizations today, business and governmental ones alike.

Schema mappings are high-level declarative specifications of the relationship between two database schemas. As mentioned earlier, they play a fundamental role in data inter-operability tasks. In particular, they are used to in particular, they are used in data exchange to specify how source data should be migrated to the target. Schema mappings are generally preferred over "lower-level" languages, such as SQL or Java, because they are more amenable to analysis, manipulation, and optimization.

In this part of the tutorial, we will review a variety of schema-mapping languages that have been studied in depth. The main focus will be on schema mappings specified by *Global-and-Local-As-View (GLAV) constraints*, which are also known as *source-to-target tuple-generating dependencies*; we will also review two important classes of GLAV constraints, namely, *Global-As-View (GAV) constraints* and *Local-As-View (LAV) constraints*. In addition, we will discuss basic concepts and methods from data exchange, including the concept of *solutions* in the context of data exchange, the

concept of *universal solutions* (i.e., the preferred solutions in data exchange), and the *chase* method for efficiently constructing universal solutions.

Understanding and Designing Schema Mappings Finally, to prepare our audience for the main topics of this tutorial, we will describe the problem of *understanding and designing schema mappings*; given a schema mapping, how can we understand what its semantics is? Conversely, given a source and target schema, how can we help a user design the desired schema mapping over these schemas?

We will review prior work with regards to these two questions. As mentioned earlier, there are several research and commercial schema-mapping design systems that provide a graphical user interface to facilitate the process of designing schema mappings. These systems typically display the source schema on one side, and the target schema on the other. Users can then specify the relationship between elements of two schemas by "drawing lines" to connect them. >From this visual specification, either a high-level schema mapping or a low-level execution script is automatically generated.

We will describe the general methodology used by such schema-mapping design systems and will assess their strengths and limitations [2]. In particular, we will discuss the inherent limitations of this approach, thus motivating the need for alternative methods for specifying schema mappings.

2.2 The Interplay between Schema Mappings and Data Examples

2.2.1 From Schema Mappings to Data Examples

A *data example* is a pair of source and target instances that conform the source and target schemas. Early uses of data examples in understanding schema mappings include the following. Data examples have been used in [17] to illustrate certain aspects of schema mappings specified by SQL queries. They have also been used to illustrate grouping semantics of schema mappings over nested source and target schemas, as well as ambiguities that may arise in a visual specification [1].

More recent work, which will be covered in considerable depth in this part of the tutorial, has focused on the problem of "capturing" schema mappings via finitely many examples. We will discuss the main results in [3]. We will first introduce the notion of a data example and describe how "good" data examples can be used to illustrate the semantics of a schema mapping. To illustrate a schema mapping, we first consider natural data examples that are finite sets of *positive* and *negative examples*. Intuitively, these are data examples that satisfy (or, resp., do not satisfy) the given schema mapping. We will show that, in general, these data examples are not sufficiently powerful to capture the full semantics of schema mappings. For this reason, we will turn to a more powerful notion of data examples, called *universal examples*.

A *universal example* is a data example in which the target instance is a universal solution for the source instance. We will explain how universal examples can be viewed as a partial specification of a schema mapping, and conversely, how a schema mapping can be semantically identified with an infinite set of universal examples. Specifically, we will introduce the concept of a *uniquely characterizing set of data examples* in this tutorial. That is, a set of data examples for which there is only one (up to logical equivalence) schema mapping that "fits" the set of data examples. Results concerning the unique characterizability of schema mappings are interesting because they can be viewed as an unambiguous way of illustrating and explaining the semantics of a schema mapping. We will describe how all LAV and n-modular schema mappings can be uniquely characterized via finitely many universal examples. In contrast, some GAV schema mappings are uniquely characterizable this way, but some other GAV schema mappings are not. We will discuss the algorithmic problem of deciding, given a GAV schema mapping, whether or not it is uniquely characterizable by finitely many universal examples. Along the way,we will unveil a tight connection between unique characterizations via universal examples to the existence of Armstrong bases (which is a relaxation of the classical notion of Armstrong databases), as well connections with constraint satisfaction and and homomorphism dualities.

Finally, we will discuss various open problems, including the following: unique characterizability for unrestricted GLAV schema mappings; other types of data examples; the number of examples needed to characterize a schema mapping; and the more conceptual problem of what it means to pick a small "representative" set of data examples when a uniquely characterizing finite set of data examples is too large or does not exist.

2.2.2 From Data Examples to Schema Mappings

So far, we have discussed how data examples can be used to illustrate or "capture" schema mappings. We now turn to another important use of data examples in schema-mapping design, namely, as a basis for deriving schema mappings. There are different scenarios where the need arises to derive a schema mapping from one or more given data examples. Each data example can be viewed as a partial specification of the intended behavior of the schema mapping on a given input instance. Automatically or semi-automatically deriving schema mappings from examples provides an alternative approach to deriving schema mappings from visual specifications, without the user having to manually spell out the schema mapping. We will discuss several proposed approaches to the problem of deriving schema mappings from data examples [12, 4, 16, 11]. We will focus on two of these approaches in depth, so as to provide the audience with an appreciation of the technical problems and existing solutions in this line of investigation.

The problem investigated in [12] is how to derive a schema mapping based on a single data example (I, J). It involves the notion of a "repair" of a schema mapping, which, intuitively, is formed by restricting and/or refining the constraints of the schema mapping in such a way that they account for every fact in the target instance J. The problem of deriving a schema mapping from a data example is then cast as an optimization problem, namely, finding a schema mapping of minimum cost, under a cost model that measures the length of the smallest repair of a schema mapping.

The second approach is based on the notion of *fitting*. A schema mapping is said to *fit* a given a set of data examples, if each of the data examples is a universal example for the schema mapping in question. In general, for a given set of data examples, there may be zero, one, or more than one fitting schema mapping. Two fundamental algorithmic problems were identified in [4]: the *Fitting Decision Problem* and the *Fitting Generation Problem*. The Fitting Decision Problem asks, for a given set of data examples, whether a fitting schema mapping exists, while the Fitting Generation Problem asks to generate a fitting schema mapping, if one exists. In [4], these problems were studied systematically for GLAV schema mappings, GAV schema mappings, and LAV schema mappings. It was shown that, whenever, for a given set of data examples, a fitting GLAV schema mapping exists, then a *most general* fitting GLAV schema mapping exists and can be efficiently constructed. The last part of the tutorial, which is detailed below, includes a discussion of the EIRENE system, which is based on this framework.

We will wrap up this discussion by highlighting various open problems, such as considering schema-mapping languages that are

more expressive than the language of GLAV constraints; this problem arises in cases where no fitting GLAV schema mappings exists.

2.3 Interactive Design of Schema Mappings via Data Examples

In this last part of the tutorial, we will cover a number of systems and frameworks for schema mapping design that use data examples in an interactive way. In particular, we will discuss the Muse and EIRENE research prototype systems, and recent work that casts example-driven schema mapping design as a learning problem.

Muse [1] is one of the earliest systems that uses data examples to facilitate schema-mapping design. Specifically, it uses data examples to identify the right schema mapping among a set of candidate schema mappings. Muse was designed to handle nested relational schemas, and it uses a schema-mapping language that goes beyond GLAV constraints and can express different grouping semantics. From a visual specification of correspondences, the system generates a collection of candidate schema mappings (taking into account different possible grouping semantics and other inherent ambiguities of visual specifications). Next, through a sequence of interactions, the set of candidate schema mappings is narrowed down to a single schema mapping. Each interaction consists of the user classifying a data example generated by the system to illustrate some aspect of ambiguity. The data examples that Muse presents to the user can be synthetic or drawn from a real database, whenever possible. We will report our experience with Muse on some publicly available schemas.

Next, we will discuss the EIRENE system, which allows a user to interactively design schema mappings via data examples. The core technique behind EIRENE is an optimized algorithm for solving the fitting problem for GLAV schema mappings. Given a finite set of data examples, the algorithm decides whether or not there exists a GLAV schema mapping that "fits" these data examples. Whenever a fitting GLAV schema mapping exists, EIRENE constructs the "most general" schema mapping that fits those data examples. When a fitting GLAV schema mapping does *not* exist for given set of data examples, EIRENE assists the user to modify the examples by identifying the tuples in the data examples that are contributing to the non-existence of a fitting schema mapping.

After this, we will explain how the interactive design of schema mappings using data examples can also be studied also using the lens of computational learning theory, as was recently proposed in [10]. We will start out by providing a gentle introduction to the relevant aspects of computational learning theory, which will include the exact learning models developed by Dana Angluin and the Probably-Approximately-Correct (PAC) learning model developed by Leslie Valiant. We will describe how the problem of deriving a schema mapping from data examples can be cast as a learning problem, in the sense of computational learning theory. Our main focus will be on the class of GAV (Global-As-View) schema mappings. We will present the main results from [10] concerning the learnability of GAV schema mappings in the exact learning model and the PAC model.

Finally, we discuss various open problems, such as the question of whether GAV schema mappings are efficiently PAC learnable with membership queries, and whether LAV and GLAV schema mappings are efficiently learnable in various learning models.

3. REFERENCES

[1] Bogdan Alexe, Laura Chiticariu, Renée J. Miller, and Wang Chiew Tan. Muse: Mapping Understanding and deSign by Example. In *ICDE*, pages 10–19, 2008.

[2] Bogdan Alexe, Wang Chiew Tan, and Yannis Velegrakis. STBenchmark: Towards a Benchmark for Mapping Systems. *PVLDB*, 1(1):230–244, 2008.

[3] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23, 2011.

[4] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Designing and refining schema mappings via data examples. In *SIGMOD*, pages 133–144, 2011.

[5] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Eirene: Interactive design and refinement of schema mappings via data examples. *PVLDB*, 4(12):1414–1417, 2011.

[6] Pablo Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[7] Philip A. Bernstein, Todd J. Green, Sergey Melnik, and Alan Nash. Implementing Mapping Composition. *VLDB Journal*, 17(2):333–353, 2008.

[8] Philip A. Bernstein and Laura M. Haas. Information integration in the enterprise. *Communications of the Association for Computing Machinery (CACM)*, 51(9):72–79, 2008.

[9] A. Bonifati, E. Q. Chang, T. Ho, V. S. Lakshmanan, and R. Pottinger. HePToX: Marrying XML and Heterogeneity in Your P2P Databases. In *VLDB*, pages 1267–1270, 2005.

[10] Balder ten Cate, Víctor Dalmau, and Phokion Kolaitis. Learning schema mappings. In *Proceedings of the International Conference on Database Theory (ICDT 2012)*, pages 22–33. ACM Press, 2012.

[11] George H. L. Fletcher, Marc Gyssens, Jan Paredaens, and Dirk Van Gucht. On the expressive power of the relational algebra on finite sets of relation pairs. *TKDE*, 21(6):939–942, 2009.

[12] Georg Gottlob and Pierre Senellart. Schema mapping discovery from data instances. *JACM*, 57(2), 2010.

[13] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *ACM SIGMOD*, pages 805–810, 2005.

[14] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *ACM PODS*, pages 61–75, 2005.

[15] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM PODS*, pages 233–246, 2002.

[16] Anish Das Sarma, Aditya G. Parameswaran, Hector Garcia-Molina, and Jennifer Widom. Synthesizing view definitions from data. In *ICDT*, pages 89–103, 2010.

[17] L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-Driven Understanding and Refinement of Schema Mappings. In *ACM SIGMOD*, pages 485–496, 2001.