

Putting Context into Schema Matching

Philip Bohannon
Bell Labs, Lucent Technologies
bohannon@research.bell-labs.com

Wenfei Fan †
University of Edinburgh & Bell Labs
wenfei@research.bell-labs.com

Eiman Elnahrawy *
Rutgers University
eimann@cs.rutgers.edu

Michael Flaster
Bell Labs, Lucent Technologies
mflaster@research.bell-labs.com

ABSTRACT

Attribute-level schema matching has proven to be an important first step in developing mappings for data exchange, integration, restructuring and schema evolution. In this paper we investigate *contextual schema matching*, in which selection conditions are associated with matches by the schema matching process in order to improve overall match quality. We define a general space of matching techniques, and within this framework we identify a variety of novel, concrete algorithms for contextual schema matching. Furthermore, we show how common schema *mapping* techniques can be generalized to take more effective advantage of contextual matches, enabling automatic construction of mappings across certain forms of schema heterogeneity. An experimental study examines a wide variety of quality and performance issues. In addition, it demonstrates that contextual schema matching is an effective and practical technique to further automate the definition of complex data transformations.

1. INTRODUCTION

A *schema mapping* is a data transformation that, given an instance of a source schema, will produce an instance that conforms to a target schema while preserving the appropriate information content of the source. Finding schema mappings is a common task in a wide variety of data exchange and integration scenarios. A *schema matching* is a pairing of attributes (or groups of attributes) from the source schema and attributes of the target schema such that pairs are likely to be semantically related. In many systems finding such a schema matching is an early step in building a schema mapping. Even with some availability of domain expertise, the computation of a schema matching may not be easy since the task itself may be large, involving dozens of tables and thousands of attributes. The combined effort of understanding an unfamiliar schema and matching it to another is a substantial burden.

As a result, automated support for schema matching has received

*Work performed in part while the author was an employee of Bell Labs.

†Supported in part by EPSRC GR/S63205/01, GR/T27433/01 and BBSRC BB/D006473/1.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.

VLDB '06, September 12-15, 2006, Seoul, Korea.

Copyright 2006 VLDB Endowment, ACM 1-59593-385-9/06/09.

$\mathcal{R}_S.inv$

id	name	type	instock	code	descr
0	leaves of grass	1	Y	0195128	hardcover
1	the white album	2	Y	B002UAX	audio cd
2	heart of darkness	1	N	0486611	paperback
3	wasteland	1	Y	0393995	paperback
4	hotel california	2	N	B002GVO	elektra cd

(a) Instance of source inventory table

$\mathcal{R}_T.book$

id	title	isbn	price	format
50	the historian	0316011770	15.57	hardcover
51	lance armstrong's war	0486400611	15.95	hardcover
52	• • •			

(b) Instance of target book table

$\mathcal{R}_T.music$

id	title	asin	price	sale	label
80	x&y	B0006L16N8	13.29	12.50	capitol
81	moonlight	B0009PLM4Y	13.49	9.99	sony
82	• • •				

(c) Instance of target music table

Figure 1: Example source and target instances

a great deal of attention in the research community (see [29] for a recent survey). In state-of-the-art schema matching systems, schema matches are discovered by considering a wide variety of evidence that may indicate a match, including similarity of data, similarity of schema and metadata information, preservation of constraints, and transitive similarity based on other known mappings (see [11, 20] for example). Once verified by the user, matches discovered by the schema matching process constitute a key input to the creation of schema mappings. In particular, the matches form the basis of constraints that should be upheld by a mapping – a valid mapping from source to target instances ensures that these constraints are enforced (see, for example, [24, 28]).

While schema matching, as described, may be challenging in itself, there are many cases where such matchings fail to capture information critical to the construction of a schema mapping. We illustrate this with an example.

Example 1.1: Consider the problem of finding a mapping between schemas \mathcal{R}_S and \mathcal{R}_T for the retail inventory tables shown in Figure 1. In the source table, $\mathcal{R}_S.inv$, information about books and CDs being sold by “Company S” is provided, and a type field indicates whether the object is a book or music. In the target schema, for “Company T”, information about books and music are stored in separate tables.

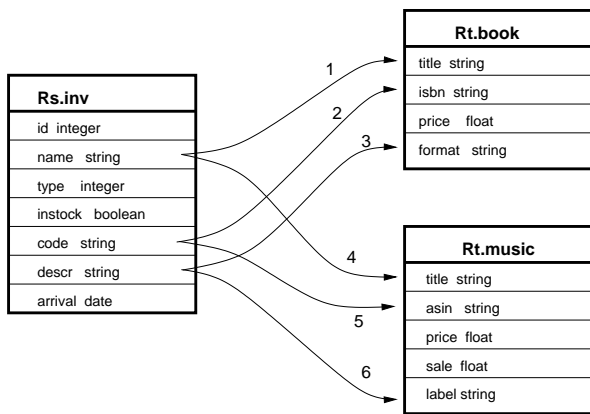


Figure 2: A traditional schema match for inv, books and music.

A traditional schema matching system might give (some subset of) the matches (numbered 1-6) between \mathcal{R}_S and \mathcal{R}_T shown in Figure 2. While this set of matches can form the basis of a schema mapping, it is ambiguous and clearly does not help the user discover the semantic distinction between the two target tables. \square

In this paper, we introduce the notion of a *contextual schema match*, in which each match is annotated with a logical condition providing the context in which the match should apply. In this example, matches 1-3 might be annotated with the condition `type = 1`, while 4-6 should hold where `type = 2`. Equivalently, one might think of views being introduced into the source or target schema to reflect a common context for several attribute matches, as shown in Figure 3.

Whenever one or more inheritance relationships are implicit in data, a database designer must choose, based on application needs, between placing the sub-types in separate tables or in a common table. This is a common-form of schema heterogeneity [29]. Example 1.1 shows one case of this since “books” and “CDs” are sub-types of “inventory items”, but other examples abound: in one school’s database both faculty and teaching assistants may appear in a single employee table, while in another school’s database separate tables may be used, and similarly for conference and journal papers in a bibliography, apartments and houses in a real-estate database, etc. Clearly, contextual schema matches directly eases the task of overcoming this form of schema heterogeneity.

Another important case where contextual matches are needed is that of “attribute normalization” in which separate rows of one table correspond to different attributes in the same row of another table, as illustrated below.

Example 1.2: Consider supplementing \mathcal{R}_S shown in Figure 1 with the $\mathcal{R}_S.\text{price}$ table appearing in Figure 4. A standard schema matching tool might find only the matching : ($\mathcal{R}_S.\text{price.price} \rightarrow \mathcal{R}_T.\text{music.price}$). However, a contextual match is more helpful, in which this match is conditioned on ($\mathcal{R}_S.\text{price.prcode} = \text{“reg”}$). Ideally, a second match ($\mathcal{R}_S.\text{price.price} \rightarrow \mathcal{R}_T.\text{music.sale}$) would be discovered based on the context ($\mathcal{R}_S.\text{price.prcode} = \text{“sale”}$). \square

From these examples, it is clear that semantically correct conditions associated with matches increase the value of those matches to the user. The additional information helps the user construct a semantically correct mapping between \mathcal{R}_S and \mathcal{R}_T .

Contributions. The identification of contextual matching as an important extension of schema matching is our first contribution. The second contribution is to define a general framework for finding good contextual matches. A salient feature of this framework is that it treats schema matching largely as a black box, and thus can

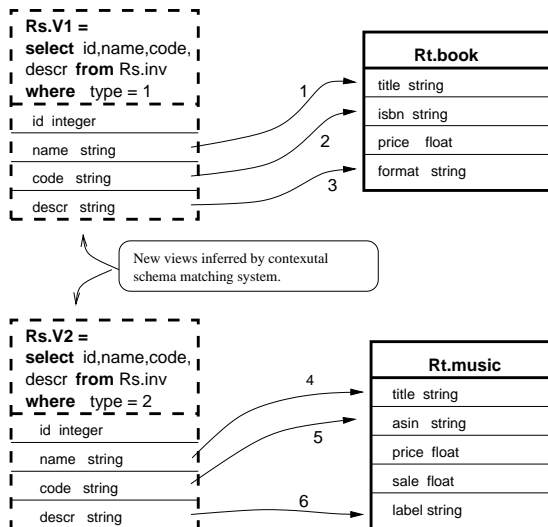


Figure 3: A contextual schema match for inv, books and music.

$\mathcal{R}_S.\text{Price}$		
id	prcode	price
0	reg	14.95
1	reg	27.99
1	sale	24.99
2	reg	8.95
2	sale	8.45
3	reg	11.40
4	sale	12.25
4	reg	14.95

Figure 4: Price Table

be used with any (instance-based) schema matching technique.

Our third contribution is the development of techniques for identifying good candidate conditions. One technique, *TgtClassInfer*, uses a classifier built on target attribute values while a second, *SrcClassInfer* depends only on *internal* features of a source table to identify promising conditions by rating some attributes on their ability to *classify* values of other attributes.

Our fourth contribution is the definition of *filtering criteria* for contextual match conditions. These filtering criteria are important since there are many *possible* contextual matches, and it is critical that, in addition to finding semantically correct contextual matches, a contextual matching algorithm does not confuse the user with too many false positives.

We next consider what happens when contextual matches are presented to an automated mapping generation algorithm like that of Clío [28, 14], in which the presence of contextual matches equates to the presence of *views* on the source or target table. In many cases, these views will be handled correctly by a Clío-style algorithm, but in general the joins necessary to construct the correct mapping for attribute normalization cannot be found with the standard rules.

Our fifth contribution is to extend the definition of the key-foreign key relationship so that *contextual key-foreign key* constraints are well-defined between views and either other views or base-tables. We also introduce rules for inferring some of these constraints based on the nature of the view.

Our sixth contribution is the definition of new join rules that, when added to Clío, allow automatic generation of a variety of mappings involving attribute normalization. We note that this extended framework would also allow Clío to better cope with a standard schema match in which views were intentionally included by

