U-MAP: A System for Usage-Based Schema Matching and Mapping[*]

Hazem Elmeleegy[†]
AT&T Labs - Research
Florham Park, NJ 07932, USA
hazem@research.att.com

Jaewoo Lee, El Kindi Rezig, Mourad Ouzzani
Purdue University
West Lafayette, IN
lee748@purdue.edu
erezig@purdue.edu
mourad@purdue.edu

Ahmed Elmagarmid
Qatar Computing Resceearch Institute - Qatar Foundation
Doha, Qatar
aelmagarmid@qf.org.qa

ABSTRACT

This demo shows how usage information buried in query logs can play a central role in data integration and data exchange. More specifically, our system U-Map uses query logs to generate correspondences between the attributes of two different schemas and the complex mapping rules to transform and restructure data records from one of these schemas to another. We introduce several novel features showing the benefit of incorporating query log analysis into these key components of data integration and data exchange systems.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases

General Terms

Algorithms, Design

Keywords

Schema mapping, schema matching, query logs

1. INTRODUCTION

Schema matching and schema mapping are essential steps in data integration and data exchange. Schema matching generates correspondences between attributes found in two different schemas. Schema mapping discovers more complex transformation rules, or *mappings*, that enable transforming and restructuring data records from one schema to another. Given the significance of the problem, a large body of literature has been devoted to address these two issues. This is evident by the surveys in [6] and [7] on schema matching, and the several systems developed for schema mappings (e.g., [1, 4, 2]).

We demo U-Map, a system that leverages the *usage* information found in *query logs* to generate attribute correspondences using the technique described in [3] and to solve some of the hard-yet-common challenges [2] facing any schema mapping tool, such as Clio [4]. We illustrate these challenges through an example from the bookstores domain (Figure 1).

The first issue is that the level of similarity between the source and target schemas may be low, in the sense that the schema structures and the attribute names are quite different. The attribute names in Figure 1 are shown to be similar across the two schemas only for ease of interpretation. Even the data values associated with the schemas may not be available in the target. This will make it very difficult for existing schema matching tools to discover the correct attribute correspondences.

The second issue is that current schema mapping tools are unable to determine IS-A relationships. More specifically, when the schema has two relations that are subclasses of a higher super class relation, and those two relations happen to be overlapping (i.e., some of their entities are shared), then we will need to merge these relations in a way that also merges the records of their shared entities. This is important so that we do not have duplicate records in the target database. For example, in the X-Schema, two relations X_Customer and X_Distributor refer to the same higher concept (Buyer for example). Following traditional algorithms, each of these relations would be mapped independently to the relation Y_Customer resulting in data duplication because records in X_Customer and X_Distributor may refer to the same real-world entity.

The third issue stems form some limitations in the classical chase algorithm, which is used to find associations, or *logical relations*. Since this algorithm only relies on chasing foreign keys in the forward direction, it might miss some *interesting associations* as we show in Section 2.

The fourth issue is that some mappings might map multiple attributes from the source schema to one attribute in the target schema, or vice versa. For example, in the X-Schema, the X_Order_Line relation references the X_Address relation more than once; once for the shipping address and once for the billing address. As a result, the logical relation for X_Order_Line would include the X_Address attributes more than once. Likewise, the logical relation for Y_Order_Line would include the address attributes more than once (billing address, shipping address and customer address). Since we

[*]This research was supported by QCRI, and by NSF Grant Numbers IIS 0916614 and IIS 0811954, and by the Purdue Cyber Center.

[†]This work was mostly done while this author was at Purdue University.

SIGMOD'11, June 12–16, 2011, Athens, Greece.
ACM 978-1-4503-0661-4/11/06.

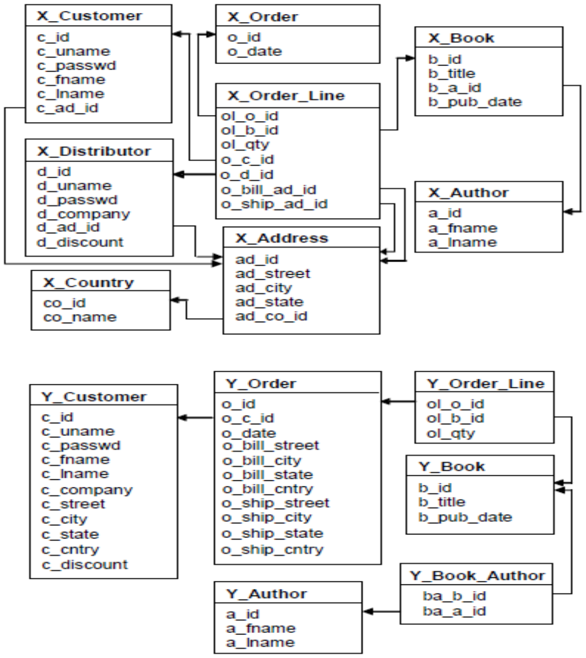


Figure 1: Schemas of two bookstores.

have more than one version for each address attribute in both logical relations, the question is: what is the correct way of mapping them?

In U-Map, we take advantage of the query logs to address all of the above issues. In the following section, we will give an overview of the main features of the U-Map system, showing how the processes of schema matching and schema mapping can benefit from the usage information in the query logs. In Section 3, we describe our demonstration scenario.

2. SYSTEM DESCRIPTION

U-Map has two main steps: Correpondences generation and mappings generation, this latter in turn includes three key steps steps. These steps are explained in the following subsections.

2.1 Correspondences Generation

Our usage-based schema matching approach operates in two main phases. It begins with a feature extraction phase, where features characterizing the usage of attributes in each schema are extracted from their respective query logs. We consider structure-level features, which reflect relationships across attributes in terms of their usage, and also element-level features, which characterize each attribute in isolation from other attributes.

The second phase is the matching phase, where attribute correspondences are generated. For each possible set of correspondences, we compute a scoring function, which reflects the level of similarity between the extracted features on both sides when the given correspondences are applied [3].

2.2 Mappings Generation

Before describing U-Map's mapping generation, we explain how a mapping specifies the transformation of records across the two schemas in Figure 1. In particular, the following expression maps every book record along with its corresponding author record in X-Schema into three different records in Y-Schema (because of the many-to-many relationship between books and authors in that schema).

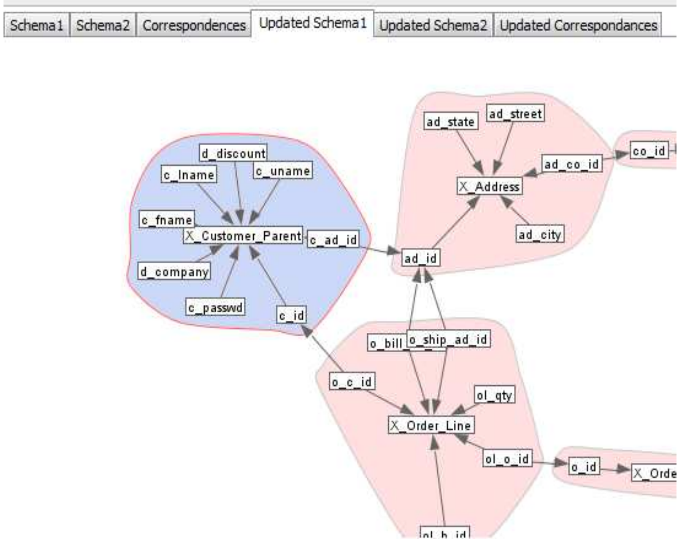


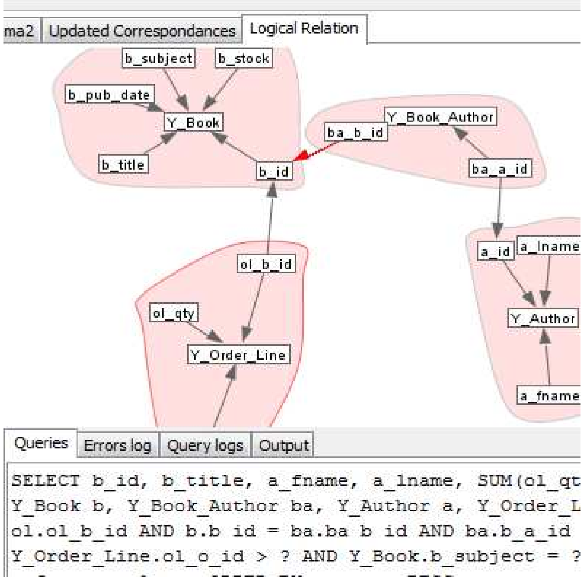
Figure 2: Part of the updated X_Schema after merging the overlapping relations

```
m1   for a in X_Author, b in X_Book
     where (a.a_id = b.b_a_id)
     ⇒ exits a' in Y_Author, b' in Y_Book, ba' in Y_Book_Author
     where (a'.a_id = ba'.ba_a_id ∧ b'.b_id = ba'.ba_b_id
     with ∧ a'.a_fname = a.a_fname ∧ a'.a_lname = a.a_lname
     ∧ b'.b_title = b.b_title ∧ b'.b_pub_date = b.b_pub_date)
```

The first two clauses in m_1, for and where, represent the source *logical relation*, which is a join between multiple source tables (or sometimes just a single table). Similarly, the second two clauses, exists and where, represent the target logical relation. Finally, the with clause represents all the correspondences between the attributes of the source and target logical relations.

Merging sibling relations: In this step, U-Map looks for relations that have IS-A relationships with a superclass in each schema. If the discovered relations are overlapping, U-Map merges them into one relation, this process is performed if one of these two criteria holds: (a) Two attributes from different relations in the source schema correspond to one attribute in the target schema. For example c_uname in X_Customer and d_uname in X_Distributor both correspond to c_uname in Y_Customer. (b) Two relations from the same schema have foreign keys referencing the same relation, this is an indication that they might have an IS-A relationship with a superclass. By looking at the the query log, U-Map determines whether two relations are overlapping or disjoint; if they are found overlapping, they are merged into one relation. Figure 2 shows the result of merging X_Customer and X_Distributor in X-Schema into X_Customer_Parent.

Constructing logical relations: U-Map extends the classical chase algorithm by using the query logs. The basic idea is that we do not always stop the chase process once there is no more foreign keys to chase. Instead, U-Map chases the relations in the opposite direction if the query logs indicate that these are meaningful relations. For example, when we



```
SELECT b_id, b_title, a_fname, a_lname, SUM(ol_qt
Y_Book b, Y_Book_Author ba, Y_Author a, Y_Order_L
ol.ol_b_id AND b.b id = ba.ba b id AND ba.b_a_id
Y_Order_Line.ol_o_id > ? AND Y_Book.b_subject = ?
```

Figure 3: The logical relations generated from the aggressive chase.

start chasing from the Y_Order_Line relation, we also include the Y_Book_Author relation even though no foreign key is referencing this relation starting from Y_Order_Line. This is because the query logs indicated that it is meaningful to include the Y_Book_Author relation in the resulting logical relation for Y_Order_Line, that's because because users were interested in joining Y_Book which is referenced by Y_Order_Line with Y_Book_Author.

U-Map lists the results of the aggressive chase algorithm which correspond to the logical relations with the added semantics extracted from the query logs. The user is given the possibility to probe these results and find out why they have been generated. Figure 3 shows part of the logical relation in U-Map which is the result of chasing from the source table Y_Order_Line, the figure also shows queries that supported the generation of this logical relation. The edge between b_id and ba_b_id represents the reverse chasing from Y_Order_Line that led to including Y_Book_Author in the logical relation because the query log of Y_Schema showed that it is meaningful to include this information along with Y_Order_Line. An example supporting this observation is shown at the bottom of Figure 3.

Creating mappings with conflict-resolution: As we mentioned earlier, the final mappings might not be conflict-free and multiple attributes in the source schema may correspond to one attribute in the target schema. This is the result of multiple foreign key references from one same relation to another relation. In our scenario, the logical relation for X_Order_Line would include the attributes of the X_Address relation more than once because there are two foreign keys referencing the X_Address relation (shipping and billing addresses). Similarly, we have multiple address attributes for the target relation Y_Order_Line as highlighted previously. U-Map uses the query logs to capture the different contexts of the conflicting attributes and then map the contexts accordingly. For example, billing address and shipping address constitute different contexts.

3. DEMONSTRATION SCENARIO

U-Map is equipped with a user-friendly graphical interface. It allows the user to provide all the inputs to the problem (i.e., schemas, query logs, and possibly correspondences); displays a graphical visualization for key objects like logical relations and mappings, as well as the schemas themselves with their correspondences; and also has a text display area for showing sample queries from the log that are related to a given logical relation or mapping. U-Map is also interactive as it allows users to modify the output of each step before proceeding to the following step. Moreover, users can configure U-Map by either enabling or disabling each one of its new features outlined in Section 2.

For attribute correspondences, the user is free to either provide them directly or request U-Map to generate them through the usage-based technique. In fact, U-Map can also generate the correspondences using a standard non-usage-based technique, which is Similarity Flooding (SF) [5].

The scenario for this demo is based on the bookstores example. We have prepared three versions of the pair of schemas shown in Figure 1 (along with their query logs), where the level of attribute name similarity is either high, medium, or low. The user will then visually compare the accuracy of the correspondences generated using each of U-Map and SF. The benefits of U-Map's usage-based technique will become clearer when the attribute name similarity is low. Next, the user will inspect the outputs of each step of the mapping generation process, once when all the new features are enabled, and once when they are disabled. In the latter case, U-Map will behave similarly to the existing schema mapping tools, which will help in highlighting the value of U-Map's new features.

4. REFERENCES

[1] Y. An, A. Borgida, R. J. Miller, and J. Mylopoulos. A semantic approach to discovering schema mapping expressions. In *ICDE*, 2007.

[2] H. Elmeleegy, A. K. Elmagarmid, and J. Lee. Leveraging query logs for schema mapping generation in U-Map. In *SIGMOD*, 2011.

[3] H. Elmeleegy, M. Ouzzani, and A. K. Elmagarmid. Usage-based schema matching. In *24th International Conference on Data Engineering*, pages 20–29, Cancún, México, April 2008.

[4] R. Fagin, L. M. Haas, M. Hernandez, R. J. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. *Conceptual Modeling: Foundations and Applications*, pages 198–236, 2009.

[5] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: a versatile graph matching algorithm. In *ICDE*, 2002.

[6] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[7] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics*, 4:146–171, 2005.