

# GAIA: a Framework for Schema Mapping Reuse (DISCUSSION PAPER)

Paolo Atzeni<sup>1</sup>, Luigi Bellomarini<sup>2</sup>, Paolo Papotti<sup>3</sup>, and Riccardo Torlone<sup>1</sup>

<sup>1</sup> Università Roma Tre, Italy,

<sup>2</sup> Banca d'Italia, Italy,

<sup>3</sup> EURECOM, France

## 1 Introduction

Schema mappings are widely used as a tool for data exchange and integration. However, although there are systems supporting data architects in the creation of mappings [5], designing them is still a time-consuming task. In this framework, given the overwhelming amount of “enterprise knowledge” stored in traditional data warehouses and in data lakes, reuse is an opportunity of increasing importance [1]. In particular, data transformation scenarios are often defined over schemas that are different in structure but similar in semantics. This is especially true if data sources, which are extremely heterogeneous, have to be mapped to a shared format. It follows that a great opportunity to reduce the effort of transformation design is to *reuse existing schema mappings*. Unfortunately, there is no obvious approach for this problem. Consider the following example.

*Example 1.* A central bank maintains a register with balance data from all companies in the country (Figure 1). This register has schema  $\mathbf{G}$ , with a relation **Balance** storing, for each company, its gains, zone of operation, and economic sector. External providers send data to the bank in different forms. Provider A adopts a schema  $\mathbf{S}_A$ , with a relation  $\mathbf{R}_A$  for companies (firms), with gains, zone of operation, and economic sector, whose code refers to relation **Activity**. Provider B adopts a schema  $\mathbf{S}_B$ , with a relation  $\mathbf{R}_B$  for companies (enterprises), their gains, sector, capital, and area, whose code refers to relation **Location**. Data is moved from  $\mathbf{S}_A$  and  $\mathbf{S}_B$  into  $\mathbf{G}$ , by using two schema mappings:

$$\sigma_A: R_A(f, g, z, s), \text{Activity}(s, d) \rightarrow \text{Balance}(f, g, z, d).$$

$$\sigma_B: R_B(e, g, s, c, a), \text{Location}(a, n) \rightarrow \text{Balance}(e, g, n, s).$$

The example shows a *data exchange scenario* where the differences in the mappings are due to the structural differences between  $\mathbf{S}_A$  and  $\mathbf{S}_B$ , which are, on the other hand, semantically very similar. Moreover, every new data provider (e.g.,  $\mathbf{S}_C$  in the figure) would require the manual design of a new, ad-hoc mapping, even if there is a clear analogy with the already defined mappings.

Our goal is to reuse  $\sigma_A$  and  $\sigma_B$  and avoid the definition of a new mapping for  $\mathbf{S}_C$ . The intuition is to collect all available mappings in a repository; then,

---

Copyright © 2019 for the individual papers by the papers authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors. SEBD 2019, June 16-19, 2019, Castiglione della Pescaia, Italy.

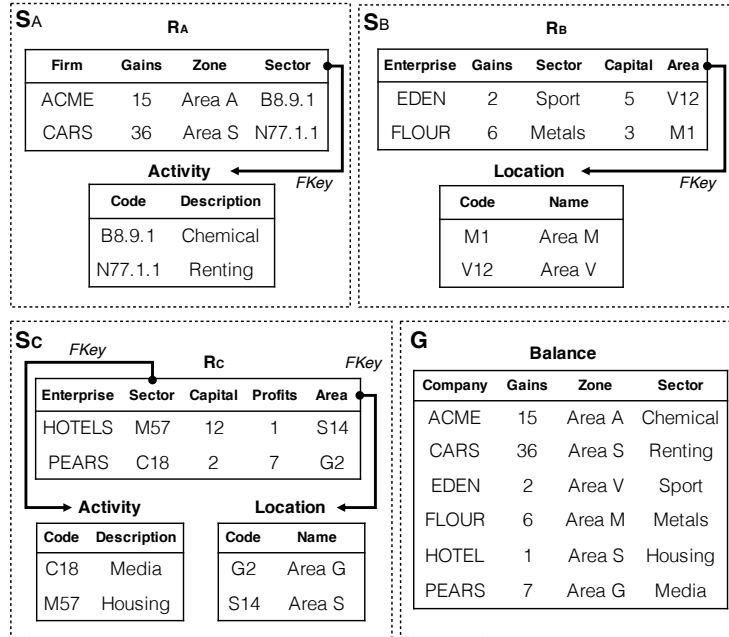


Fig. 1. A data transformation scenario

for any new pair of schemas (e.g.,  $\mathbf{S}_C$  and  $\mathbf{G}$  in the figure), query such repository to retrieve a suitable mapping. Unfortunately, this form of direct reuse is complicated by the nature of schema mappings. A mapping characterizes the constraint between a pair of schemas at a level of detail that enables both logical reasoning and efficient execution. Yet, a simple variation in a schema, such as a different relation or attribute name or a different number of attributes, makes it not applicable. Our experiments show that mappings from a corpus of 1,000 schema mappings can be reused for new, unmapped pairs of schemas only in 20% of cases. To be reusable, a mapping should be described in a way that is independent of its specificities but, at the same time, harnesses the essence of the constraint so as to work for similar schemas.

*Example 2.* Consider a “generic” mapping  $\Sigma_A$ , obtained from  $\sigma_A$  by replacing names of the relations and attributes with variables. It could be informally described as follows:

$\Sigma_A$ : for each relation  $r$  with key  $f$  and attributes  $g, a, s$   
for each relation  $r'$  with key  $s$  and attribute  $d$   
with a foreign key constraint from  $s$  of  $r$  to  $s$  of  $r'$   
there exists a relation  $r''$  with key  $f$  and attributes  $g, a, d$ .

If instantiated on  $\mathbf{S}_A$ , the generic mapping  $\Sigma_A$  expresses a mapping to  $\mathbf{G}$  that is the same as  $\sigma_A$ . This solution seems a valid compromise between precision, i.e., the ability to express the semantics of the original mapping, and generality,

as it can be applicable over different schemas. However,  $\Sigma_A$  falls short of the latter requirement, as it is not applicable on  $\mathbf{S}_B$ . Indeed, there are no constraints on attribute  $g$  and  $a$ , and so they could be bound to any of **Gains**, **Sector** and **Capital**, incorrectly trying to map **Capital** into the target.

*Example 3.* Consider now a more elaborated generic mapping that uses constants to identify attributes:

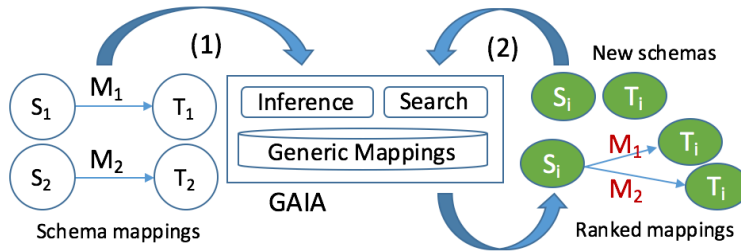
$\Sigma_B^H$ : for each relation  $r$  with key  $e$  and attributes  $g, s, c, a$   
 for each relation  $r'$  with key  $a$  and attribute  $d$   
 with a foreign key constraint from  $a$  of  $r$  to  $a$  of  $r'$   
 where  $g = \text{Gains}$ ,  $s \neq \text{Gains}$ ,  $s \neq \text{Capital}$ ,  $c \neq \text{Gains}$ ,  $c \neq \text{Sector}$   
 there exists a relation  $r''$  with key  $e$  and attributes  $g, d, s$ .

This generic mapping is precise enough to correctly describe both  $\sigma_A$  and  $\sigma_B$  and can be re-used with other schemas.

The example shows a combination of attributes, identified by constraints on their names and role, that form a correct and useful generic mapping. Once pinpointed, generic mappings can be stored in a repository, so that it is possible to use them for a new scenario. In our example, given  $\mathbf{S}_C$  and  $\mathbf{G}$ , the generic mapping  $\Sigma_B^H$  can be retrieved from the repository and immediately applied.

There are three main challenges in this approach.

- We need a clear characterization of what it means for a generic mapping to correctly describe and capture the semantics of an original schema mapping.
- As a generic mapping is characterized by a combination of conditions on attribute names and roles, for a given schema mapping there is a combinatorial number of generic mappings. We need a mechanism to generate them.
- For a new scenario (e.g., new schemas), there is an overwhelming number of generic mappings that potentially apply, with different levels of “suitability”. We need efficient tools to search through and choose among them.



**Fig. 2.** The architecture of GAIA.

In this work, we address the above challenges with GAIA, a system for mapping reuse. GAIA supports two tasks, as shown in Figure 2: (1) infer generic mappings, called *meta-mappings*, from input schema mappings, and store them in a

repository; (2) given a source and a target schema, return a ranked list of meta-mappings from the repository which are used to generate possible mappings between these schemas. GAIA provides the following key contributions:

- The notion of *fitness*: a semantics to precisely characterize and check when a meta-mapping is suitable for a reuse scenario.
- An algorithm to *infer* meta-mappings from schema mappings with an approach that extends previous efforts for the definition of schema mappings by example; this algorithm is used to populate a repository of meta-mappings supporting schema mapping reuse.
- An approach to reuse based on: (i) the search, in the repository of available meta-mappings, for those that *fit* a new pair of source and target schemas and (ii) the construction, from the retrieved meta-mappings, of possible mappings to be proposed to the designer.

Because of space limitation, algorithms and details are in the full version of the paper [2]. In the rest of this paper, we provide examples of meta-mappings (Section 2) and experimental results from an evaluation of our system with more than 20,000 real-world data transformations over 40,000 schemas (Section 3).

## 2 Mapping and meta-mappings

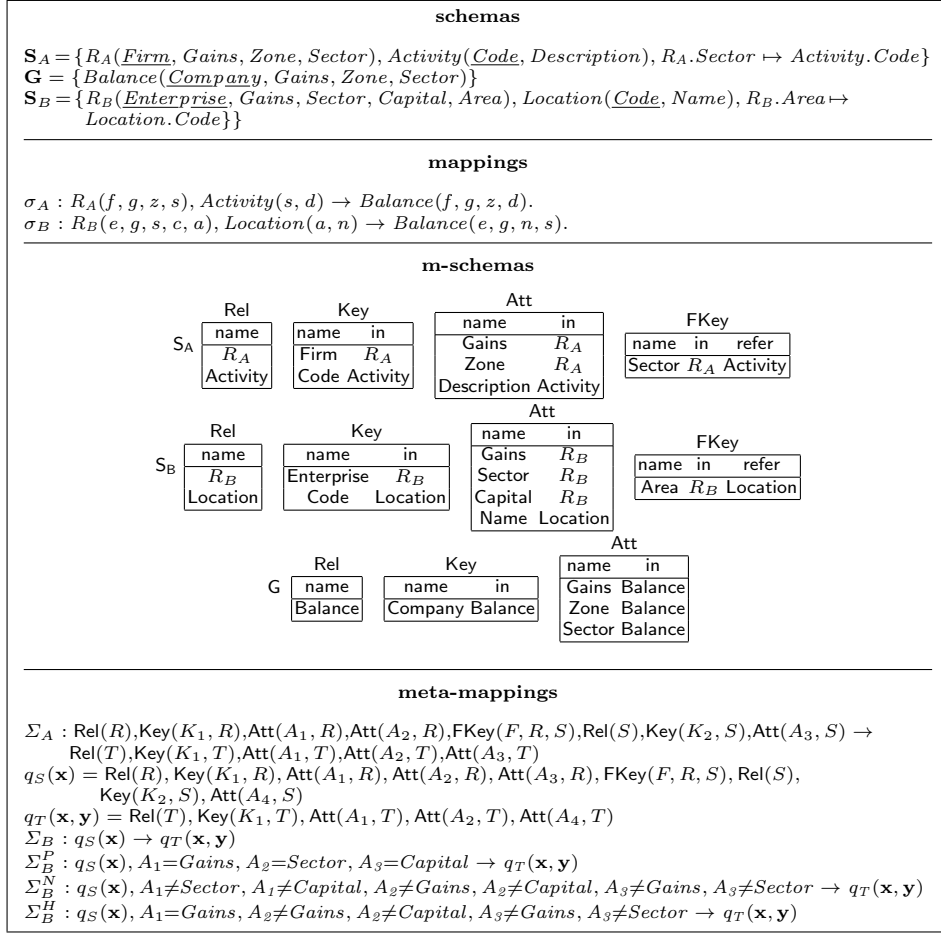
We recall the notion of schema mapping [6] and introduce that of *meta-mapping*. While the former notion models specific transformations, the latter introduces an abstraction over mappings [7, 8] and models *generic* mappings between schemas. Building on these notions, we illustrate the functionalities of our system.

**Schema mappings.** Let  $\mathbf{S}$  (the source) and  $\mathbf{T}$  (the target) be two relational schemas and let  $Inst(\mathbf{S})$  and  $Inst(\mathbf{T})$  denote the set of all possible instances of  $\mathbf{S}$  and  $\mathbf{T}$ , respectively. A (*schema*) *mapping*  $M$  for  $\mathbf{S}$  and  $\mathbf{T}$  is a binary relation over their instances, that is,  $M \subseteq Inst(\mathbf{S}) \times Inst(\mathbf{T})$  [3].

Without loss of generality, we consider mappings expressed by a single *source-to-target tuple-generating-dependency* (st-tgd)  $\sigma: \forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$  where  $\mathbf{x}$  and  $\mathbf{y}$  are two disjoint sets of variables,  $\phi(\mathbf{x})$  (the left-hand-side, LHS) is a conjunction of atoms involving relations in  $\mathbf{S}$  and  $\psi(\mathbf{x}, \mathbf{y})$  (the right-hand-side, RHS) is a conjunction of atoms involving relations in  $\mathbf{T}$ . The dependency represents a mapping  $M$  in the sense that  $(I, J) \in M$  if and only if  $(I, J)$  satisfies  $\sigma$ . In this case,  $J$  is called a *solution* of  $I$  under  $\sigma$ . We can compute a suitable  $J$  in polynomial time by applying the *chase procedure* to  $I$  using  $\sigma$  [6]: the result may have labeled nulls denoting unknown values and is called the *universal* solution, since it has a homomorphism to any possible solution  $J'$ , that is, a mapping  $h$  of the nulls into constants and nulls such that  $h(J) \subseteq J'$ .

*Example 4.* Consider the schemas  $\mathbf{S}_A$ ,  $\mathbf{S}_B$  and  $\mathbf{G}$  of Figure 1, which we recall in Figure 3 with all the formalisms that will be discussed throughout the paper.

A mapping between  $\mathbf{S}_A$  and  $\mathbf{G}$  is the st-tgd  $\sigma_A$  discussed in the Introduction and reported in Figure 3 (quantifiers are omitted for the sake of readability).



We assume, hereinafter, that, given a schema, its corresponding m-schema is also given, and vice versa. A meta-mapping is expressed by means of an st-tgd over dictionaries that describes how the elements of a source m-schema map to the elements of a target m-schema.

*Example 5.* Mapping  $\sigma_A$  of Example 4 can be expressed, at the dictionary level, by meta-mapping  $\Sigma_A$  in Figure 3. This st-tgd describes a generic transformation that takes two source relations  $R$  and  $S$  linked by a foreign key  $F$  and generates a target relation  $T$  obtained by joining  $R$  and  $S$  on  $F$  that includes: the key  $K_1$  and the attributes  $A_1$  and  $A_2$  from relation  $R$  and the attribute  $A_3$  from  $S$ .

Given a source m-schema  $\mathbf{S}$  and a meta-mapping  $\mathcal{M}$ , a target m-schema  $\mathbf{T}$  is generated by applying the chase procedure to  $\mathbf{S}$  using  $\mathcal{M}$ .

*Example 6.* The chase of  $\mathbf{S}_A$  using  $\Sigma_A$ , both in Figure 3, generates the following target m-schema where  $\perp_R$  is a labelled null denoting a relation name.

Rel	Key	Att
name	name in	name in
$\perp_R$	$Firm \perp_R$	$Gains \perp_R$
		$Zone \perp_R$
		$Description \perp_R$

This m-schema describes the relational schema:  $R(\underline{Firm}, Gains, Zone, Description)$

A meta-mapping operates at schema level rather than at data level and thus provides a means for describing generic transformations. Subtleties could arise from the chase procedure in presence of existential quantifications in meta-mappings producing duplications of relations in the result. This is avoided by assuming that, for each existentially quantified variable, there is a target *equality generating dependency* (egd) [6] ensuring that whenever two relations in the target have the same structure, then they coincide.

**From meta-mappings to mappings.** Given a source schema  $\mathbf{S}$  and a meta-mapping  $\Sigma$ , it is possible not only to generate a target schema by using the chase, as shown in Example 6, but also to automatically obtain a schema mapping  $\sigma$  that represents the *specialization* of  $\Sigma$  for  $\mathbf{S}$  and  $\mathbf{T}$  [8]. The *schema to data exchange transformation* (SD transformation) generates from  $\mathbf{S}$  and  $\Sigma$  a complete schema mapping made of  $\mathbf{S}$ , a target schema  $\mathbf{T}$  (obtained by chasing the m-schema of  $\mathbf{S}$  with the meta-mapping), and an s-t tgd  $\sigma$  between  $\mathbf{S}$  and  $\mathbf{T}$ . The correspondences between LHS and RHS of  $\sigma$  are derived from the provenance information computed during the chase step, in the same fashion as the provenance computed over the source instance when chasing schema mappings [4].

*Example 7.* Consider again the scenario in Figure 3. If we apply the SD transformation to the schema  $\mathbf{S}_A$  and the meta-mapping  $\Sigma_A$ , we obtain the target m-schema of Example 6 and the following mapping from  $\mathbf{S}_A$  to  $\perp_R$ :

$$\sigma : R_A(f, g, z, s), Activity(s, d) \rightarrow \perp_R(f, g, z, d).$$

Thus, we get back, up to a renaming of the target relation, the mapping  $\sigma_A$  in Figure 3 from which  $\Sigma_A$  originates.

**From mappings to meta-mappings** While previous work focused on generating a mapping from a given meta-mapping [8], we tackle the more general problem of *mapping reuse*, which consists of: (i) generating a repository of meta-mappings from a set of user-defined schema mappings, and (ii) given a new pair of source and target schemas, generating a suitable mapping for them from the repository of meta-mappings.

*Example 8.* For the scenario in Figure 3, our system first generates several fitting meta-mappings from  $\sigma_A$  between  $\mathbf{S}_A$  and  $\mathbf{G}$ . Once  $\mathbf{S}_B$  and  $\mathbf{G}$  are given as input for a new transformation, the system scans the corpus of existing meta-mappings and identifies  $\Sigma_A$  as a fitting meta-mapping from  $\mathbf{S}_B$  to  $\mathbf{G}$ . This meta-mapping is then instantiated to generate a schema mapping between them, according to the SD transformation above.

### 3 Experimental evaluation

We implemented GAIA in PL/SQL 11.2 for Oracle 11g. All experiments were conducted on Oracle Linux, with an Intel Core i7@2.60GHz, 16GB RAM.

**Datasets and Transformations.** We used data transformations that are periodically executed to store data coming from several data sources into the *Central National Balance Sheet* (CNBS) database, an archive of financial information of about 40K enterprises. The schema of CNBS has 5 relations with roughly 800 attributes in total. Source data come from three different providers. The Chamber of Commerce provides data for 20K companies (datasets **Chamber**). While the schemas of these datasets are similar, the differences require one mapping per company between relations of up to 30 attributes with an average of 32 (self) joins in the LHS of the mappings. Data for 20K more companies is collected by a commercial data provider in a single database (**CDP**). This database is different both in structure and attributes names from the schemas in **Chamber** and requires one mapping involving 15 joins in its LHS. Finally, data for further 1,300 companies (**Stock**) is imported from the stock exchange, with each company requiring a mapping with 40 joins on average.

**Transformation scenarios.** For the inference of the meta-mappings, we consider two configurations: *single*, where a meta-mapping is inferred from one mapping, and *multiple*, where the inference is computed on a group of mappings for companies in the same business sector. We observe that the results stabilize with 10 schema mappings in the group and do not improve significantly with larger numbers, therefore consider 10 mappings for the *multiple* configuration.

**Search precision.** Let  $\sigma$  be a mapping from a source  $\mathbf{S}$  to a target  $\mathbf{T}$  and let  $\mathcal{Q}$  be the set of meta-mappings inferred from  $\sigma$ . We measure, in terms of *search precision*, the ability of the system to return the meta-mappings in  $\mathcal{Q}$  when queried with  $\mathbf{S}$  and  $\mathbf{T}$ , i.e., how well the system retrieves correct cases.

We use a *resubstitution* approach: we populate an initially empty repository by inferring the meta-mappings from a set  $\Theta$  of mappings. For each schema mapping  $\sigma$  in  $\Theta$ , we will denote by  $\mathcal{Q}_\sigma$  all the meta-mappings inferred from

$\sigma$ . We then pick a schema mapping  $\sigma'$  from  $\Theta$  and search the repository by providing as input the source and target schemas of  $\sigma'$ . We then collect the top-10 results according to the coverage and compute the percentage of correctly retrieved meta-mapping, that is, those that belong to  $\mathcal{Q}_{\sigma'}$ .

We test search precision with a corpus of mappings of increasing size (from 200 to 21,301). In each test, we query the repository of meta-mappings inferred from the given mappings by using 50 different source-target pairs and report their average search precision. The experiment is executed on single and multiple configurations. The largest repository contains about 700K explicit meta-mappings in single configuration and 110K in multiple configuration. In the multiple scenario, the test is considered successful on a mapping  $\sigma$  when the retrieved meta-mapping originates from the group that includes  $\sigma$ . On average, a meta-mapping includes 12 equality and/or inequality constraints.

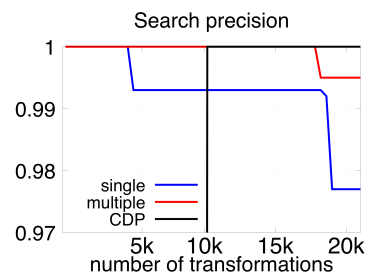


Fig. 4. Search precision.

The plot in Figure 4 shows that search precision decreases with the size of the repository as larger numbers of mappings lead to an increasing number of false positives in the result. We report the search precision for the **CDP** mapping with a separate line. When this mapping is inserted in the repository (after 10K transformations), it is immediately identified with both configurations. This shows that specific structures and constants

lead to meta-mappings that are easy to retrieve. Meta-mappings derived from multiple schema mappings improve search precision because are more general than the ones from single mappings, thus reducing the risk of overfitting.

## References

1. D. Abadi et al. The Beckman report on database research. *Commun. ACM*, 59(2):92–99, Jan. 2016.
2. P. Atzeni, L. Bellomarini, P. Papotti, and R. Torlone. Meta-mappings for schema mapping reuse. *PVLDB*, 12(5):557–569, 2019.
3. P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *SIGMOD*, 2007.
4. L. Chiticariu and W. C. Tan. Debugging schema mappings with routes. In *VLDB*, pages 79–90, 2006.
5. R. Fagin, L. M. Haas, M. A. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling*, 2009.
6. R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, 2003.
7. M. A. Hernández, P. Papotti, and W. C. Tan. Data exchange with data-metadata translations. *PVLDB*, 1(1):260–273, 2008.
8. P. Papotti and R. Torlone. Schema exchange: Generic mappings for transforming data and metadata. *Data Knowl. Eng.*, 68(7):665–682, 2009.