Data integration with many heterogeneous sources and dynamic target schemas (extended abstract)

Luigi Bellomarini, Paolo Atzeni, Luca Cabibbo

Università Roma Tre, Italy

1 Introduction and motivation

Information integration is the general problem that arises in applications that need to consolidate (in a virtual or materialized way) data coming from different sources [9, Ch.21]. In this paper, we consider a scenario for data integration, very common in practice, for which existing solutions are not effective. We refer to those applications where there are many (dozens or hundreds or even more) sources, in the same domain, that have to contribute to one, single global ("target") system. A common case is that of "central" organizations that receive data from a large set of "local" companies or administrations; a specific case is that of a national central bank that receives data from all the banks in the country. This scenario is often handled by imposing to all local sources an exchange format, so that data are transferred to the central institution in a standardized form. In some cases, this solution is just inapplicable, as companies might refuse the adoption of the standard unless forced by regulations. Moreover, exchange formats (especially for statistics and finance) are inherently flexible, versatile and unstable, allowing a number of different source schemas. Finally, due to the complexity of exchange formats, their adoption is often partial or incorrect. In particular, a diverging interpretation of the exchange format can be even accepted in case the central institution realizes that the various sources have specific features that, though not general, are nevertheless interesting and deserve to appear in the target.

An approach based on the theory of mappings [6] can be interesting in this case, but the large quantity of source schemas would render the classical techniques not much effective, as it would require the specification of many different mappings, between each of the various sources and the target. Also, the sources are not completely known beforehand (and there are often new ones, also similar, but with additional differences) and therefore mappings cannot be directly specified in advance. Finally, given the interest in considering specific features of sources, it turns out that even the target schema should not be fixed in advance, as some portions depend on sources (their schema and even their data). All these three aspects are not addressed by current mapping-based approaches.

We also have an observation that mitigates the difficulties: in many cases the sources are indeed different from one another, but they do share similarities that can be exploited.

On the basis of the above requirements we propose a new approach, where there is one source schema S_0, used as a reference, and the other source schemas can be seen as variations of S_0. Then, we consider variations on the target schema, induced by specific features of the source schemas, including their data, in order to support "schematic transformations" [11], that is, the possibility to generate schema elements in the final target schema G, given data and schema elements in S_0.

G

RepositoryOfBalances

Bank	*IoBName*	*Amount*
1005	Asset	8
1005	Liability	19
1006	Asset	-42
1006	Liability	21

G'

RepositoryOfBalancesAlt

Bank	*Asset*	*Liability*
1005	8	19
1006	-42	21

Fig. 1. Sample target data

For example, Figure 1 represents two desired global schemas, each of which is able to store summaries of balance sheets of banks. The relation RepositoriesOfBalances stores the *Amount* (that is, the difference between credits and debits) aggregated by *Bank* and *IoBName* (item of balance name). RepositoriesOfBalancesAlt ("Alt" stands for "alternate") stores the same items for each bank with the difference that *Asset* and *Liabilities* are attributes.

Figure 2 represents the data as they are collected from the sources. S_0 is the established exchange format, where BalSheetTemplate prescribes the local institution to structure its balance in relations (one or many) having the attributes *Year*, *Bank*, *IoBName*, *IoBCred*, the amount of credit for the item of balance, *IoBDeb*, the amount of debit for the item of balance.

In the example, S_1 is identical to S_0 and reports the balance sheet as a single relation. In S_2 the balance sheet is (horizontally) partitioned into two relations, with the same attributes as S_0: tuples having *Year* preceding 2012 are stored in BalSheet2010-2011, the others in BalSheet2013. In S_3, the balance is (vertically) decomposed into credits and debits, with a coarser level of granularity, as data are reported and grouped by *Bank*, disregarding the years.

In the rest of the paper we study the data integration scenario we have just sketched. In Section 2 we show how we handle a large number of heterogeneous but similar sources and in Section 3 we discuss how we cope with the issue of generating new features in target schemas on the basis of source schemas and data. Finally, in Section 4 we draw our conclusions and briefly discuss related work.

S_0

BalSheetTemplate

Year	*Bank*	*IoBName*	*IoBCred*	*IoBDeb*

S_1

BalSheet

Year	*Bank*	*IoBName*	*IoBCred*	*IoBDeb*
2010	1005	Asset	35	27
2011	1005	Liability	29	10
2010	1006	Asset	41	30
2010	1006	Liability	31	30
2013	1006	Asset	0	53
2013	1006	Liability	33	13

S_2

BalSheet2010-2011

Year	*Bank*	*IoBName*	*IoBCred*	*IoBDeb*
2010	1005	Asset	35	27
2011	1005	Liability	29	10
2010	1006	Asset	41	30
2010	1006	Liability	31	30

BalSheet2013

Year	*Bank*	*IoBName*	*IoBCred*	*IoBDeb*
2013	1006	Asset	0	53
2013	1006	Liability	33	13

S_3

BalSheetCredit

Bank	*IoBName*	*IoBCred*
1005	Asset	35
1005	Liability	29
1006	Asset	41
1006	Liability	64

BalSheetDebit

Bank	*IoBName*	*IoBDeb*
1005	Asset	27
1005	Liability	10
1006	Asset	83
1006	Liability	43

Fig. 2. Sample source data

2 Handling many sources

Let us first concentrate on cases in which the global schema is completely defined beforehand (and so coinciding with the baseline global schema G_0), while in Section 3, we give some insights about how we handle dynamic global schemas.

We recognize that each source schema S_i can be considered as a variation of S_0. In this, we notice an analogy with the *schema evolution* problem [3, 8], where schemas are derived from one another through the application of simple and standardized operations, namely schema evolution operators.

Thus, we propose a new approach where each S_i is described as an evolution of S_0, that is, as if it were the result of a "virtual" application of one or more schema evolution operators to S_0. In the remainder of the paper we will concentrate only on transformations involving a single operation. We consider a small set of operators, which is however sufficiently expressive and general for most common real scenarios: addition/deletion of attributes, partition of tuples into separate relations, union of relations into a single relation, projection decomposition of a relation into multiple ones, join of multiple relations into a single one, attribute dereferencing (that is extracting an attribute into a foreign key-related relation).

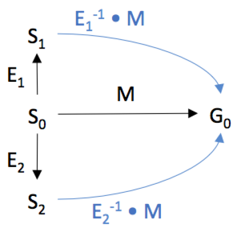

Fig. 3. Mappings for the scenario of variations

2.1 Transformations

Transformations could be modeled in various ways, procedural or declarative. We adopt the notion of schema mapping [4], where the relationships between source and target schemas are described in terms of first-order *s-t tgd* (*source-to-target tuple generating dependencies*). [1]

Hence, each transformation relating the reference schema S_0 to any S_i can be associated with a schema mapping E_i. In the example, S_1 is identical to S_0 (so we have the identity mapping), while S_2 is obtained with a partition on the basis of *Year* values and S_3 with decomposition by projection.

Then, in our example, the mappings between S_0, S_2 and S_3 would be represented as follows:

E_2: BalSheet(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d), $y < 2012$
$\rightarrow$ BalSheet2010-2011(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d);

BalSheet(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d), $y \geq 2012$
$\rightarrow$ BalSheet2013(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d)

E_3: BalSheet(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d)
$\rightarrow$ BalSheetCredit(Bank: b, IoBName: i_n, IoBCred: sum(i_c, groupBy(b, i_n))),
BalSheetDebit(Bank: b, IoBName: i_n, IoBDeb: sum(i_d, groupBy(b, i_n)))

The idea we have just discussed, that the various source schemas are variations of a reference one S_0, allows us to concentrate on the mapping between S_0 and the target schema G_0. This is summarized in Figure 3, where M relates S_0 to G_0 and E_i is the representation of the transformation between S_0 and a specific source S_i. Then, with our approach, the user should focus just on the reference source schema S_0 and describe its mapping M to the global schema. Indeed, we are interested in mappings between the various S_i's and G. Intuitively, this can be obtained by composing the inverse of each E_i with M. Let us go back to our example; with respect to G in Figure 1, we have:

[1] Our tgd's are indeed a bit more complex than those usually found in data exchange settings [4], but their semantics can be defined as an extension of the classical one. Specifically they contain scalar operations (for example a difference of values) and aggregations (like *sum* with *group by* in E_3) in the rhs [2].

M: BalSheet(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d)
$\rightarrow$ RepoOfBalances(Bank: b, IoBName: i_n, Amount: sum($i_c - i_d$, groupBy(b, i_n)))

where for each *Bank* and *IoBName*, the *Amount* is calculated by subtracting debits from credits. The aggregation then sums the contributions from different *Year*s. In order to make M work also for S_2, we should "undo" the transformations from S_0 to S_2 (E_2) and then apply the original mapping M, as summarized in Figure 3. In other words, the goal would be to find a new mapping $M_2^* = E_2^{-1} \circ M$ and similarly for every other possible source S_i. In most cases, the various results would need to be consolidated, usually by means of a *merge* operator [3], for which there can be various versions whose details are not relevant here.

2.2 Technical issues

Let us now discuss the technical issues. Indeed, our goal is to find a mapping M_i^* between S_i and G_0, given the mappings E_i between S_0 and S_i and M between S_0 and G_0. For this kind of problems, solutions have been discussed in the literature, based on *composition* and *inversion* operators [6, 7][2]. However, such solutions are not sufficient for our case as they would require the existence of an "exact inverse" (a *maximum recovery* [1]) for E_i.

One major problem is that rarely do schema mappings have such an exact inverse, as they may involve information loss; in such cases, source instances cannot be rebuilt from target ones. This is also the case for our transformations: E_3, for instance, is lossy, since it decomposes a relation into two other relations where aggregations are applied. This causes loss of information and the original relation cannot be reconstructed with a join. Vice versa, E_2 is lossless, because the partition (which, by definition, forbids overlapping) can be reversed with a union. However, let us consider, in this case, a relaxed notion of inverse, for example *quasi-inverse*s of schema mappings [7] [3]. It does not exactly rebuild the original instance, but another one, affected by information loss. For example a quasi-inverse of E_3 is

E_3^{-1}: BalSheetCredit(Bank: b, IoBName: i_n, IoBCred: i_c),
BalSheetDebit(Bank: b, IoBName: i_n, IoBDeb: i_d)
$\rightarrow \exists y$ (BalSheet(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d))

Notice that E_3^{-1} is far from being an exact inverse because the aggregated contributions of different *Year*s have not been decomposed; moreover *Year* is existentially quantified and its original values have not been restored. Nevertheless, we deem that this inverse is sufficient for our purpose since, in this case, E_3 loses less information than M. In facts, with a simple substitution, the composition

[2] The inverse M^{-1} of a schema mapping M is such that $M^{-1} \circ M = \text{Id}$, where Id is the identity mapping, transforming each instance into itself.

[3] Notice that the original definition does not foresee aggregations, which however we support here.

with M yields:

$$\begin{aligned}
E_3^{-1} \circ M:\quad & \text{BalSheetCredit(Bank: } b\text{, IoBName: } i_n\text{, IoBCred: } i_c\text{),}\\
& \quad \text{BalSheetDebit(Bank: } b\text{, IoBName: } i_n\text{, IoBDeb: } i_d\text{)}\\
\rightarrow\ & \text{RepoOfBalances(Bank: } b\text{, IoBName: } i_n\text{, Amount: sum(}i_c - i_d\text{, groupBy}(b, i_n)\text{))}
\end{aligned}$$

An interesting idea is the definition of some kind of order relation $\preceq_s$, based on the amount of "transferred information" [1]: $M \preceq_{S_0} E_i$ holds if M transfers less information than E_i when applied to the same source S_0. In this case we observe that $M \preceq_{S_0} E_3$; indeed it is possible to verify this by observing that the instances of G can be generated from the ones of S_3.

Our result is that whenever the order relation $M \preceq_{S_0} E_i$ holds for every S_i, it is possible to integrate all the differing sources, also in presence of aggregations, calculating M_i^* through a suitable notion of inverse, for example quasi-inverses. An intuitive proof for this consists in verifying that the order relation guarantees the presence of some mappings from S_i to G (since E_i transfers more information than M) and one of them, M_i^*, can be calculated by suitably inverting E_i and composing with M.

3 Dynamic target schemas

Let us now consider the more general case, where the global schema is not defined in advance, but it may depend on the specific sources. Different scenarios are indeed possible and the global schema may depend on source data or metadata (names of relations and attributes) or both. Here we concentrate on the most interesting case, which is the one where attributes of the global schema derive from values in the sources. This is also the most relevant condition in real contexts, as it can be the consequence of a common practice in exchange formats between a local and a central organization, that of embedding "schematic" information into data.

An example appears in Figure 2, where the values of *IoBName* ("Asset" and "Liability") are schematic elements as they qualify the other two attributes *IoBCred* and *IoBDeb*: a credit or a debit, in facts, is meaningful only if referred to a specific item of balance. An alternative representation would have four attributes, for expressing credits and debits for each of the two possible item names. In this latter representation, the introduction of a new item of balance, for instance "Equity", would require to alter the schema giving rise to a new variant and so on.

Here we sketch a novel technique to handle such situations. Basically, it consists in the definition of *template tgd's*, an extended version of usual tgd's that allow metadata-data correspondences. They are intended to be used for expressing correspondences between S_0 and the baseline global schema G_0, in the mapping M. We allow quantified variables not only to denote values, but also attribute names. Indeed, the baseline global schema referred to in the rhs of template tgd's is somehow polymorphic, as it contains variables for the attributes that are not known in advance, but their presence and name depend on the source data.

A key feature of our approach is a *rewriting algorithm*, generating usual tgd's out of the template ones, given the source data. For each tuple matching the lhs of a template tgd, the algorithm generates one or more attributes in the schema of the rhs, eventually specifying the global schema G, where all the attributes have been made explicit. The tgd's generated in this way respond to the usual definition and can be then enforced with the common *chase* procedure [5].

An example of template tgd is the following mapping M' between S_0 and G_0:

M': BalSheet(Year: y, Bank: b, IoBName: i_n, IoBCred: i_c, IoBDeb: i_d)
$\rightarrow$ RepoOfBalancesAlt(Bank: b, i_n: sum($i_c - i_d$, groupBy(b)))

The rhs describes the baseline global schema G_0, with an attribute variable, i_n, used for unknown attributes. For its value, the expression sum(i_c−i_d, groupBy(b)) specifies that in RepositoryOfBalancesAlt, *Asset* and *Liability* are all calculated as aggregations (an aggregation each), grouping by the only disaggregated attribute, which is *Bank*.

Initially G' contains all the attributes that are fully specified in G_0 (only *Bank* in this case). In the rewriting phase, for each tuple in the lhs, an attribute named after the value bound to i_n ("Asset" or "Liability") is added. [4] At the end, the rewritten tgd, which can be chased and enforced in the usual way, is:

$\widehat{M'}$: BalSheet(Year: y_a, Bank: b, IoBName: "Asset", IoBCred: i_{ca}, IoBDeb: i_{da}),
BalSheet(Year: y_l, Bank: b, IoBName: "Liability" , IoBCred: i_{cl}, IoBDeb: i_{dl})
$\rightarrow$ RepoOfBalancesAlt(Bank: b, Asset: sum($i_{ca} - i_{da}$, groupBy(b)),
Liability: sum($i_{cl} - i_{dl}$, groupBy(b)))

4 Conclusions and related work

We provided a solution to the problem of data integration in contexts where there are a large number of different sources whose schema is not completely known beforehand and the global schema can depend on the source data. We consider a single data source as a reference and map it, with schema mappings, into the global schema. All the other sources are described as if they were obtained as a schema evolution of the reference.

While data integration has been studied in a variety of theoretical and practical contexts [12], approaches to handling differences between schemas and representing transformations with mappings are more typically studied at higher level in the model management literature, for example in [3]. More recently, the schema evolution problem has been pursued in [8, 1], however without any connection to the data integration problem or adoption of a declarative relaxed

[4] Some variable renamings are also needed to avoid unwanted matches.

notion of inverse. Theoretical details about inversion and composition are presented in [6], while a formal definition of quasi-inverse schema mappings can be found in [7].

Dynamic global schemas rely on the notion of "schematic transformation", introduced in [11]. Some known approaches are specifically oriented to *query answering* and concentrate on defining appropriate extensions to SQL or relational algebra to have result sets with a schema that varies depending on input data [11, 14]. Others use schematic information to provide a certain degree of schema independence to queries [13]. Since our specific target is data integration, we proposed an extension to the common language of mappings. A similar operation has also been done in [10], with an approach more oriented towards *data exchange* and the goal of solving specific issues such as nesting.

References

1. M. Arenas, J. Perez, J. L. Reutter, and C. Riveros. Foundations of schema mapping management. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 227–238, 2010.
2. P. Atzeni, L. Bellomarini, and F. Bugiotti. Exlengine: Executable schema mappings for statistical data processing. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 672–682, New York, NY, USA, 2013. ACM.
3. P. A. Bernstein. Applying model management to classical meta data problems. In *CIDR Conference*, pages 209–220, 2003.
4. R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.
5. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
6. R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, 30(4):994–1055, 2005.
7. R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Quasi-inverses of schema mappings. *ACM Trans. Database Syst.*, 33(2), 2008.
8. R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Schema mapping evolution through composition and inversion. In *Schema Matching and Mapping*, pages 191–222. 2011.
9. H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book.* Prentice-Hall, Englewood Cliffs, New Jersey, second edition, 2008.
10. M. A. Hernández, P. Papotti, and W. C. Tan. Data exchange with data-metadata translations. *PVLDB*, 1(1):260–273, 2008.
11. L. V. S. Lakshmanan, F. Sadri, and S. N. Subramanian. Schemasql: An extension to sql for multidatabase interoperability. pages 476–519, 2001.
12. M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.
13. U. Masermann and G. Vossen. Sisql: Schema-independent database querying (on and off the web). In *IDEAS*, pages 55–64, 2000.
14. C. M. Wyss and E. L. Robertson. Relational languages for metadata integration. *ACM Trans. Database Syst.*, 30(2):624–660, 2005.