# Automatic Schema Merging Using Mapping Constraints Among Incomplete Sources

Xiang Li Christoph Quix David Kensche Sandra Geisler
Informatik 5 (Information Systems)
RWTH Aachen University
52056 Aachen, Germany
{lixiang,quix,kensche,geisler}@dbis.rwth-aachen.de

## ABSTRACT

Schema merging is the process of consolidating multiple schemas into a unified view. The task becomes particularly challenging when the schemas are highly heterogeneous and autonomous. Classical data integration systems rely on a mediated schema created by human experts through an intensive design process.

In this paper, we present a novel approach for merging multiple relational data sources related by a collection of mapping constraints in the form of P2P style tuple-generating dependencies (tgds). In the scenario of data integration, we opt for minimal mediated schemas that are complete regarding certain answers of conjunctive queries. Under Open World Assumption (OWA), we characterize the semantics of schema merging by properties of the output mapping system between the source schemas and the mediated schema. We propose a merging algorithm following a redundancy reduction paradigm and prove that the output satisfies the desired logical properties. Recognizing the fact that multiple plausible mediated schemas may co-exist, a variant of the a priori algorithm is employed to enumerate alternative mediated schemas. Output mappings in the form of data dependencies are generated to support the mediated schemas, which enables query processing. We have evaluated our merging approach over a collection of real world data sets, which demonstrate the applicability and effectiveness of our approach in practice.

## Categories and Subject Descriptors

H.2.1 [**Database Management**]: Logical Design—*Schema and subschema*; H.2.5 [**Database Management**]: Heterogeneous Databases; H.2.4 [**Database Management**]: Systems—*Relational databases*

## General Terms

Algorithms, Design

## Keywords

schema merging, data integration, model management, schema mappings

## 1. INTRODUCTION

Modern data intensive applications often involve a multitude of heterogeneous data sources. Schema merging is the process to consolidate multiple related heterogeneous schemas to provide a unified user view called the *mediated schema*. In order to support data loading from the sources to the mediated schema (e.g., in data warehousing), or to enable querying of sources through the mediated schema (e.g., in data integration [24] or dataspaces [31]), mappings revealing the relationship between the mediated schema and the source schemas have to be established in the merging process. Classical data integration systems [24] nowadays still rely on a mediated schema created by an intensive manual design process by human experts, which is costly and inflexible in a dynamic evolving environment such as dataspaces.

In vision of the importance of schema merging, Merge is proposed as one of the major operators in *Model Management* [7]. Nevertheless, as retrospected by Bernstein and Melnik in [9], the original vision of Model Management 1.0 is not semantic but structural, i.e., not relating schema and data. In other words, operators are interpreted in terms of schemas, while lacking connection to the underlying data of the schemas. A semantic merging approach is in need, not only for the sake of expressiveness but also for executability reasons. Merging using logical schema mappings, e.g., tgds, is inevitable for realizing a model management engine to address real-world data programmability problems. In order to be executable, a semantic merging algorithm not only consumes data dependencies as input, but also produces data dependencies as output, so that query processing or data migration can be performed.

Although differing a lot in terms of mapping language, semantics, data models, and methodologies, most existing schema merging techniques are binary, i.e., merging two schemas at a time. We deem a native n-ary merge as interesting for generating mediated schemas for a multitude of data sources. Though binary merging algorithms can be applied iteratively to merge multiple schemas, the process need a full human supervision, i.e., in each iteration an expert is required to generate mapping between a new source schema and the intermediate merged result from previous steps. Moreover, in a scenario of ad hoc P2P environments, only some particular mappings are available and nobody has the complete knowledge to produce arbitrary mappings. Therefore, a native n-ary merging algorithm is more suited to exploit all the available mappings for multiple data sources. In the survey by Batini et al. [6], they contribute the popularity of binary merge to a complexity reason, that is, by involving less input the problem of merging will become less complicated. However, as we have already described, constructing schema mappings is also quite expensive and requires a lot of human supervision, which compensates for the increased computation efforts of a native n-ary merge.

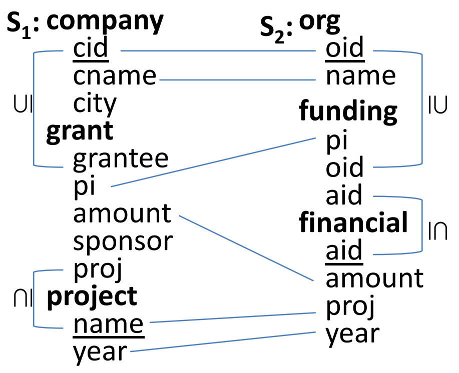

**Figure 1: The Running Example**

In this paper, we propose a native n-ary semantic merging approach. Without human supervision, it generates a series of candidate minimal mediated schemas that are complete regarding certain answers of conjunctive queries. It takes source integrity constraints and P2P style tgds as input and generates logical output mappings in the form of data dependencies. Our contributions are as follows:

- we provide a logical characterization of the semantics of schema merging based on OWA. The adoption of OWA is essential because real world data sources are independently developed, and their extensions, i.e., explicitly stored data, usually do not conform to any inter-schema logical constraints;
- we employ tuple generating dependencies among multiple schemas as the input mapping language, instead of being confined to source-to-target tgds. This extends the expressiveness of mapping languages used in [11, 10, 28]. For example, transitive closures can now be expressed in an input mapping;
- source integrity constraints in the form of tgds and egds are seamlessly incorporated into our framework;
- we provide a native n-ary algorithm for creating mediated schemas and mappings satisfying the desired semantics.
- we provide a procedure to enumerate all the mediated schemas with no redundant column; and, finally,
- we report the implementation of our approach and the evaluation results on several real world data sets and a generated workload.

As a running example, consider the schemas illustrated in Fig. 1, adapted from [18]. Given are two schemas, each one with three relations. In schema $S_1$, *grant* represents a many-to-many relationship between companies and projects. Similarly, *funding* represents a many-to-many relationship between *organization* and *financial* in $S_2$. The corresponding foreign key constraints for these relationships are indicated by $\subseteq$-lines, while keys are underlined. In addition, we show the value correspondences of the attributes of the schemas in straight lines.

The relationship between source schemas can be captured precisely in tgds:

$$M_1 : company(Cid,Cname,City) \leftrightarrow org(Cid,Cname)$$
$$M_2 : grant(G,Pi,A,Sp,Pj) \wedge project(Pj,Yr) \leftrightarrow$$
$$funding(Pi,G,Aid) \wedge financial(Aid,A,Pj,Yr)$$
$$M_3 : grant(G,Pi,A,Sp,Pj) \wedge project(Pj,Yr) \leftrightarrow$$
$$financial(Aid,A,Pj,Yr)$$
$$M_4 : project(Pj,Yr) \rightarrow financial(Aid,A,Pj,Yr)$$

Tgd $M_1$ states that the relations *company* and *org* are equivalent, except for the additional column *city* in *company*. Such simple one-to-one correspondences can be represented in most other approaches, too. However, more complex relationships can only be formalized by mappings involving joins between several relations. For example, the tgd $M_2$ states that the join of *grant* and *project* in $S_1$ is equivalent to the join of *funding* and *financial* in $S_2$. $M_3$ is similar but it states that the relation *financial* is equivalent to a projection of the join of *grant* and *project*. With the inclusion of $M_3$, we know that there are no dangling tuples of *financial* that do not join with some tuple of *funding*. Finally, $M_4$ states that all tuples from *project* are contained in *financial*. The details of the mapping language will be given in Section 2.

As in the running example, source schemas usually contain important information in the form of integrity constraints, such as keys and foreign keys. Integrity constraints reveal inner structure of a schema and hence are a significant source of information which should be taken into consideration in schema merging.

The remaining of the paper is organized as follows. Sec. 2 describes some background definitions. We characterize the semantics of schema merging in the concrete scenario of designing a mediated query interface in Sec. 3. The main aspects of our approach and the algorithms are presented in Sec. 4. Experimental results are reported in Sec. 5. Related works are covered in Sec. 6, before we conclude and discuss future work in Sec. 7.

## 2. PRELIMINARIES

**Data Dependencies:** A *tuple generating dependency (tgd)* [1], is a query containment constraint in the form of: $\forall\vec{X}[\exists\vec{Y}\phi(\vec{X},\vec{Y}) \rightarrow \exists\vec{Z}\psi(\vec{X},\vec{Z})]$, where $\phi$ and $\psi$ are conjunctions of atoms and $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ are mutually disjoint variables. It is *full*, if there are no existential variables on the right hand side, otherwise it is an *embedded* tgd. An *equality-generating dependency (egd)* [1] has the form $\forall\vec{X}[\phi(\vec{X}) \rightarrow (X_i = X_j)]$, where $\phi(\vec{X})$ is a conjunction of atoms, and $X_i$, $X_j$ are variables in $\vec{X}$. A set of tgds is said to be *weakly acyclic*, if there is no recursive implication of existential variables [19]. In the following, we omit the quantifiers for brevity. We also write $\phi(\vec{X},\vec{Y}) \leftrightarrow \psi(\vec{X},\vec{Z})$ to denote both, $\psi(\vec{X},\vec{Z}) \rightarrow \phi(\vec{X},\vec{Y})$ and $\phi(\vec{X},\vec{Y}) \rightarrow \psi(\vec{X},\vec{Z})$ in a mapping. For a source schema $S$ and a target schema $T$, a *source-to-target tgd (s-t tgd)* is a tgd such that the antecedent contains only atoms from $S$ and the consequent contains only atoms from $T$. For a schema with relations $r_1, r_2, \ldots, r_n$, a set of tgds are called *copy tgds*, if they send each source relation to a distinct target relation with the same arity, i.e., in the form of $\{\hat{r}_i(\vec{X}) \leftarrow r_i(\vec{X})\}$ with $\hat{r}_i$ being the corresponding target relation for $r_i$. For a relation $r$ with arity $n$, we say an *identity query* is a query $q(X_1,X_2,\ldots,X_n) \leftarrow r(X_1,X_2,\ldots,X_n)$, with $n$ different variables.

**Schemas and Mappings:** A *schema* $S$ is a sequence of relation symbols $(r_1,r_2,\ldots,r_n)$ where each $r_i$ has a fixed arity. An *instance* $I$ of a schema $S$, denoted by $I \in Inst(S)$, is the union of relation instances over $r_i$ where $r_i \in \mathbf{S}$. An instance $I$ is *legal* wrt. a set of data dependencies $\Sigma$ formulated as tgds and egds [1] over $S$, if it satisfies the dependencies, i.e., $I \models \Sigma$. A binary *mapping* between two schemas $S_1$ and $S_2$ is a triple $\mathscr{M} = (S_1, S_2, \Sigma)$ where $\Sigma$ is a set of dependencies. The semantics of a binary mapping is a binary relation with instances of the source schema as the domain and instances of the target schema as the range, i.e., $Inst(\mathscr{M}) = \{(I,J) : I \in Inst(S_1) \wedge J \in Inst(S_2) \wedge (I,J) \models \Sigma\}$, The semantics of the composition of two mappings $\mathscr{M}_{13} = \mathscr{M}_{12} \circ \mathscr{M}_{23}$ is then defined as $Inst(\mathscr{M}_{13}) = \{(I,K) : \exists J\ (I,J) \in Inst(\mathscr{M}_{12}) \wedge (J,K) \in Inst(\mathscr{M}_{23})\}$. For a mapping $\mathscr{M}$ from $S$ to $T$, the possible worlds, called *solutions*, of $T$ wrt. an instance $I$ of $S$ are $Sol_{\mathscr{M}}(I) = \{J \in Inst(T) : (I,J) \in Inst(\mathscr{M})\}$. We denote it by $Sol(I)$ when the mapping involved is clear from context. A solution is called a *universal solution* if there is a homomorphism from it to any other solution [19]. A universal solution that has no homomorphism to a subset of itself is called a *core* [20]. It is known that the core is unique up to isomorphism when it exists.

**Chase and Certain Answers:** The chase procedure [13] is an indispensable tool for reasoning with data dependencies. Let $\Sigma$ be a set of tgds and egds with terminating chase, we use $chase_\Sigma(I)$ to denote the result of the chase using $\Sigma$ over a database $I$. When $\Sigma$ is a set of s-t tgds and target dependencies. We also use $chase_\Sigma(I)$ to denote the target instance $J$ such that $(I,J) = chase_\Sigma(I,\emptyset)$. For a schema mapping specified by a finite set of s-t tgds $\Sigma_{st}$ and a set of target dependencies $\Sigma_t$ consisting of a finite set of weakly acyclic tgds and a finite set of egds, it is shown in [19] that for any source instance $I$, chasing $I$ against $\Sigma_{st} \cup \Sigma_t$ yields a universal solution if the chase succeeds. The *certain answer* of a query $q$ wrt. a set of database instances $P$ over the same schema is: $certain(q,P) = \bigcap_{I \in P} q(I)$. Equivalence of two sets of databases is then defined using certain answers. Two sets of database instances $P_1$ and $P_2$ under the same schema are said to be *L-equivalent* wrt. a query language class $L$, denoted by $P_1 \equiv_L P_2$, if for any query in $L$ they have the same certain answer. In this paper, we are particularly interested in equivalence wrt. conjunctive queries, i.e., CQ-equivalence.

## 3. TOWARDS A MEDIATED QUERY INTERFACE

In [26], Miller et al. first use formal schema equivalence results to characterize the semantics of schema merging. However, as we understand it today, the semantics of a mediated schema is better characterized using not only the structure of the schema itself but also the mapping relating the mediated schema to data sources that host extensional data. Based on such an observation, the semantics of schema merging in our approach is characterized as logical properties of the output mapping system between the source schemas and the mediated schema, targeting at the scenario of creating a mediated query interface over incomplete data sources. We first state the problem of n-ary schema merging in Section 3.1, where the inter-schema mapping constraints and source integrity constraints are unified. In Section 3.2, we propose the completeness criteria aiming at retaining not only all the ground data, but also certain answers ensured by data dependencies. Section 3.3 formalizes the desiderata that data asserted to be equivalent in the input mapping should be integrated via the output mapping system. In Section 3.4, we formulate minimality of a mediated schema such that no column in the schema is redundant.

### 3.1 N-ary Schema Merging

Consider the scenario of merging multiple data sources to create a single unified query interface. Since the data sources are independently developed, their extensions, i.e., explicitly stored data, usually do not conform to any inter-schema logical constraints. That is, under Closed World Assumption, logical constraints usually cannot be asserted among data sources. This is one reason why Pottinger and Bernstein [28] interpret their input mapping as specification of overlap between schemas instead of direct logical constraints over data instances. In this paper, we take the OWA by interpreting input mapping as constraints that are expected to hold over the integrated global database, which is also inline with the common assumption in data integration that sources are sound but incomplete.

*Definition 1.* Given an incomplete database $I$ of schema $S$, the *semantics* of $I$ wrt. a set of data dependencies $\Sigma$ over $S$ is $Sem_\Sigma(I) = \{I' : I' \in Inst(S) \wedge I \subseteq I' \wedge I' \models \Sigma\}$. When the dependencies are clear from context, we simply write $Sem(I)$ for brevity.

Consider $n$ source schemas $S_1, S_2, \dots, S_n$ with no two relations sharing the same name. Each source schema $S_i$ has a set of integrity constraints $\Sigma_i$ as a union of tgds and egds. The input mapping is specified by a set of inter-schema tgds $\Sigma_{in}$ among the source schemas.

*Definition 2.* The *joint source schema* is the disjoint union of the $n$ source schemas, while a *joint source instance* is the disjoint union of $n$ instances with one instance for each source schema. The *merge input* is then a pair $(S,\Sigma)$, with $S$ being a joint source schema and $\Sigma = \Sigma_{in} \cup \bigcup_i \Sigma_i$. The semantics of a joint source instance $I$ is then $Sem_\Sigma(I)$.

In the following of the paper, we only consider sound data sources, that is, they are consistent with the specified data dependencies and the possible world set they present is not empty.

*Definition 3.* For a merge input $(S,\Sigma)$, a merge output is a binary mapping $\mathscr{M}_o = (S, G, \Sigma_o)$, called the *output mapping*, in which $G$ is the *mediated schema* and $\Sigma_o$ is a set of data dependencies.

It is now commonly observed that for a given merging scenario, there may be multiple plausible mediated schemas [12, 31]. Without human intervention, our approach will produce a series of plausible outputs.

### 3.2 Completeness

In the survey by Batini et al. [6], completeness of a mediated schema is described as containing all concepts in the union of the source schemas' application domains, which is rather representative of the approaches they surveyed, since they mostly employ a conceptual model such as EER and do not explicitly consider querying of the mediated schema.

Pottinger and Bernstein [28] concretize Hull's notion of query dominance [23] in the scope of schema merging as retaining in the mediated schema the data of each source relation. We extend their formalization in two aspects. First, we consider retaining answers of all conjunctive queries besides each source relation. Second, we take source incompleteness into consideration and hence require the retainment of both, extensively stored data and inferred data, for which we use the notion of certain answers.

*Definition 4.* Given a merge input $(S,\Sigma)$, an output mapping $\mathscr{M}_o = (S, G, \Sigma_o)$ is *complete* wrt. a mapping language $L$, if there exists a mapping $\mathscr{M}_w = (G, S, \Sigma_w)$ with $\Sigma_w$ specified in $L$ such that for any joint source instance $I$, we have: $Sem_\Sigma(I) \equiv_{CQ} Sol_{\mathscr{M}_o \circ \mathscr{M}_w}(I)$. $\mathscr{M}_w$ is called a *witness mapping*.

The requirement of completeness ensures that an output mapping system has a mapping backward to the joint source schema, which is a witness of the retainment of certain answers of queries. If the witness mapping allows CQ rewriting, e.g., specified in s-t tgds, then for each $q \in CQ$ over $S$, the rewriting against $\mathscr{M}_w$ produces a query over the mediated schema producing the same certain answer.

EXAMPLE 1. *Consider a subset of our example containing only the relations company and org and the mapping* $M_1$ : $company(Cid, Name, City) \leftrightarrow org(Cid, Name)$. *The mediated schema* $\{company\_org(cid, Name, city)\}$ *is complete with the output mapping:*

$$company\_org(Cid, Name, City) \leftarrow org(Cid, Name)$$
$$company\_org(Cid, Name, City) \leftarrow company(Cid, Name, City)$$

*A witness mapping is:*

$$company(Cid, Name, City) \leftarrow \quad company\_org(Cid, Name, City)$$
$$org(Cid, Name) \leftarrow \quad company\_org(Cid, Name, City)$$

## 3.3 Integratedness

Completeness alone is not sufficient to indicate the quality of an output mapping system, as illustrated in the following example.

EXAMPLE 2. *Consider the same input as in Ex. 1. A copy of the source schema with the following output mapping is also complete:* $company'(Cid, Name, City) \leftarrow company(Cid, Name, City)$ *and* $org'(Cid, Name) \leftarrow org(Cid, Name)$. *A possible witness mapping is:*

$$company(Cid, Name, City) \leftarrow company'(Cid, Name, City)$$
$$company(Cid, Name, City) \leftarrow org'(Cid, Name)$$
$$org(Cid, Name) \leftarrow company'(Cid, Name, City)$$
$$org(Cid, Name) \leftarrow org'(Cid, Name)$$

*A problem with the above output mapping system is that, although in the input mapping* company *and* org *are asserted to be equivalent when projected to* cid *and* name*, their images in the mediated schema are still separate. This phenomenon reveals that completeness does not guarantee that equivalent data are integrated in the mediated schema.*

When creating a mediated query interface, semantically equivalent data should be provided in a seamless way in the merged schema, while the users do not need to concern about either the origin or the structural heterogeneity. We formalize below the requirement that integration is actually performed via the output mapping.

*Definition 5.* Given a merge input $(S, \Sigma)$, an output mapping $\mathscr{M}_o = (S, G, \Sigma_o)$ is *integrated* if for any joint source instance $I$, the following holds:

$$Sol_{\mathscr{M}_o}(I) \equiv_{CQ} \bigcup_{J \in Sem(I)} Sol_{\mathscr{M}_o}(J).$$

Besides mixing equivalent data, integratedness has another implication under OWA. Since data sources are incomplete, two different joint source instances may have the same semantics, i.e., the same possible world set. Integratedness requires that the output mapping does not distinguish these equivalent joint sources regarding query answering.

EXAMPLE 3. *Consider a source instance* $I = \{company(1, IBM, Armonk), org(2, Microsoft)\}$ *for Example 2. The query* $q(Y) \leftarrow company(X, Y, Z)$ *will give only IBM as certain answer. However, the same query has* $\{IBM, Microsoft\}$ *as the certain answer for* $\bigcup_{J \in Sem(I)} Sol_{\mathscr{M}_o}(J)$. *Therefore, it is not integrated. It is straightforward to verify that Example 1 is integrated.*

We have to point out that integratedness alone is not sufficient either, since a constant output mapping producing a constant target side always satisfies integratedness. Hence, integratedness has to be used together with completeness.

## 3.4 Minimality

In the scenario of creating a mediated query interface for data integration systems, we make the assumption that a smaller query interface (still retaining all the query capabilities) is better and head for a minimal schema with no redundant column. Redundancy of columns is defined wrt. a given output mapping.

*Definition 6.* For a target schema $G$ in an output mapping $\mathscr{M}_o$, the *induced mapping* wrt. a projection $\mathscr{M}_p$ of $G$ is the composition $\mathscr{M}_o \circ \mathscr{M}_p$.

An induced mapping is actually an adaptation of the output mapping after removing some columns from the mediated schema.

EXAMPLE 4. *Consider a fragment of the running example with the source schema S consisting of the relations* grant(grantee,pi,amount,sponsor), funding(pi,oid, aid) *and* financial(aid,amount,project,year). *The input mapping is*

$$grant(O, Pi, A, S) \leftarrow funding(Pi, Aid, O) \wedge financial(Aid, A, Pj, Yr).$$

*Let G be a replica of S and denote the predicates in G by* grant', funding' *and* financial'. *A complete and integrated output mapping* $\mathscr{M}_o$ *consists of the following dependencies:*

$$grant(O, Pi, A, S) \rightarrow grant'(O, Pi, A, S)$$
$$financial(Aid, A, Pj, Yr) \wedge$$
$$funding(Pi, O, Aid) \rightarrow grant'(O, Pi, A, S)$$
$$funding(Pi, O, Aid) \rightarrow funding'(Pi, O, Aid)$$
$$financial(Aid, A, Pj, Yr) \rightarrow financial'(Aid, A, Pj, Yr)$$

*Let* $\mathscr{M}_p$ *be a projection that removes the* pi *column from* grant'. *The subschema* $G'$ *contains* grant", financial" *and* funding". *The induced mapping is* $\mathscr{M} = (S, G', \Sigma)$ *with* $\Sigma$ *represented by the following dependencies:*

$$grant(G, Pi, A, S) \rightarrow grant''(G, A, S)$$
$$financial(Aid, A, Pj, Yr) \wedge funding(Pi, O, Aid) \rightarrow grant''(O, A, S)$$
$$funding(Pi, O, Aid) \rightarrow funding''(Pi, O, Aid)$$
$$financial(Aid, A, Pj, Yr) \rightarrow financial''(Aid, A, Pj, Yr)$$

*Definition 7.* An output mapping $\mathscr{M}_o = (S, G, \Sigma_o)$ is *minimal* wrt. a set of output mappings $\mathscr{P}$, if for any projection of $G$ that is not an identity, the induced mapping does not belong to $\mathscr{P}$.

The set of output mappings $\mathscr{P}$ may be the set of all mappings having certain properties, e.g., the properties *completeness* and *integratedness* as introduced above. In the following, we are always interested in the minimal mappings that are complete and integrated and simplify say minimal or minimality.

Intuitively, an output mapping is minimal, if removing any columns will cause loss of some property (e.g., completeness).

EXAMPLE 5. *Consider the induced mapping* $\mathscr{M}$ *in Example 4. It is no longer complete because the information over pi of grants is lost due to the projection. In fact, the original output mapping* $\mathscr{M}_o$ *is minimal wrt. completeness.*

The next proposition suggests we only need to focus on completeness for induced mappings of an integrated output mapping.

PROPOSITION 1. *Any induced mapping of an integrated output mapping is also integrated.*

For instance, the induced mapping in Example 4 is still integrated.

## 4. A LOGICAL FRAMEWORK

In this section, we describe an algorithm producing complete and integrated mediated schemas that have no redundant columns. The algorithm consists of two stages: a constraint repairing stage and a mediated schema minimization stage. Sec. 4.1 describes a constraint repairing procedure which produces an initial mediated schema with an output mapping that is both, complete and integrated, and which we call the canonical mediated schema. In Sec. 4.2, we provide a procedure for testing completeness after removing columns from the canonical mediated schema schema (or equivalently, whether some columns are redundant), which is at the core of the mediated schema minimization stage described in Sec. 4.3. Finally, we describe the complete algorithm in Sec. 4.4.

### 4.1 The Canonical Mediated Schema

In this section we show that there always exists a merge output that is both, complete and integrated, which we call the *canonical output mapping*. The mediated schema is called the *canonical mediated schema*. Given a merge input $(S,\Sigma)$, we create the canonical output mapping as follows:

1. Initialize $G$ to be a replica of $S$. Let $\Sigma_{copy}$ be the copy tgds, i.e., the 1-to-1 mapping from $S$ to $G$.
2. Let $\rho$ be the renaming of predicates from $S$ to their images in $G$. Construct the target dependencies over $G$ as $\Sigma_G = \rho(\Sigma)$.
3. The canonical output mapping is $\mathscr{M}_o = (S, G, \Sigma_{copy} \cup \Sigma_G)$.

The output mapping consists of two parts: a set of s-t tgds transferring all the facts in the source database to the global schema ($\Sigma_{copy}$) and a set of target dependencies over the mediated schema ($\Sigma_G$).

LEMMA 1. *The canonical output mapping $\mathscr{M}_o$ described above is both complete (wrt. full s-t tgds) and integrated.*

PROOF. Completeness: for any source instance $I$, we have $\rho(Sem(I)) = Sol_{\mathscr{M}_o}(I)$ with $\rho$ being the renaming as in the creation of the canonical output mapping. Using a set of copy tgds (the inverse of $\Sigma_{copy}$, which exists as $G$ is a 1-1 copy of $S$) as the witness mapping, we obtain a solution set co-initial (i.e., with the same minimal elements regarding set inclusion) with $Sem(I)$. Co-initiality implies CQ-equivalence, since CQ is monotone.

Integratedness: for each complete ground source $I' \in Sem(I)$ we have $\rho(I')$ as the minimal solution of $Sol_{\mathscr{M}_o}(I')$ in terms of set inclusion, because $\rho(I')$ is a solution and it is a subset of any other solution. Therefore, $\bigcup_{I' \in Sem(I)} Sol_{\mathscr{M}_o}(I')$ and $\rho(Sem(I))$ are co-initial. Considering $\rho(Sem(I))$ is identical to $Sol_{\mathscr{M}_o}(I)$, integratedness is straightforward. □

Interestingly, Melnik describes in [25] a straightforward algorithm creating a mediated schema for view integration, which differs from our canonical mediated schema only in the direction of the output mapping. However, the size of the schema does not matter for view integration and hence he does not head for minimization of mediated schemas.

EXAMPLE 6. *The output mapping $\mathscr{M}_o$ in Example 4 is the canonical output mapping obtained from the input using the procedure described.*

We have shown that a complete output mapping is able to be expressed in the language of s-t tgds and target dependencies. The following proposition reveals that without the target dependencies in the output mapping, completeness cannot be achieved.

PROPOSITION 2. *There exists an input mapping specified as a finite set of full tgds, such that no output mapping defined as a finite set of s-t tgds is complete wrt. s-t tgds.*

PROOF SKETCH. Consider the following input mapping, in which relations from source schemas $S_1$ and $S_2$ are labeled with corresponding subscripts:

$$\begin{aligned} e_1(x,y) &\rightarrow c_2(x,y) \\ c_1(x,y) \wedge e_1(y,z) &\rightarrow c_2(x,z) \\ c_1(x,y) &\leftrightarrow c_2(x,y) \end{aligned}$$

$e_1$ represents the edges in a graph, $c_1$ and $c_2$ express the transitive closure of $e_1$. A query asking for the certain answers of $c_1$ is a witness of the incompleteness, which simply follows from the inability of first order queries to express transitive closure [1]. □

### 4.2 Test Schema Reduandancy

We introduce in Section 4.2.1, a procedure testing whether the answer of a CQ is still obtainable after removing some columns from a schema. This is then employed by the algorithm in Section 4.2.2, which tests schema redundancy.

#### *4.2.1 Query Recoverability*

We first introduce in this section a useful notion which we call *query recoverability*. Intuitively, it is a formalization that the answer of a query is still obtainable after a projection.

*Definition 8.* For a given projection mapping $\mathscr{M}_p$ and a given query class $L$:

- A query $q$ is recoverable wrt. a database $I$, if $\exists q' \in L$ over the projection of $I$, producing the same answer as $q$ over $I$. We say $q$ is recoverable via $q'$.
- A query $q$ is recoverable wrt. a set of databases $P$, if $\exists q' \in L$, such that $\forall I \in P$, $q$ is recoverable via $q'$.
- A query is recoverable wrt. a set of dependencies $\Sigma$, if it is recoverable wrt. the set of all legal databases under $\Sigma$.

EXAMPLE 7. *Consider two relations* company(cid, cname, city) *and* org(oid, name) *with the dependencies $\Sigma$ that contain a tgd $company(Cid,Cname,city) \rightarrow org(Cid,Cname)$ and an egd $org(Oid,Name_1) \wedge org(Oid,Name_2) \rightarrow Name_1 = Name_2$. Let $\mathscr{M}_p$ be the projection that projects out the cname column of company. Then the query $\pi_{cname}(company)$ is recoverable wrt. $\Sigma$. If we denote the projected relation by* company'(cid, city)*, the query $\pi_{cname}(company' \bowtie org)$ can be used to recover the original answer to $\pi_{cname}(company)$.*

We describe in Figure 2 a test for conjunctive query recoverability. The correctness is given in Theorem 1. By freezing a query as a database, we mean sending each distinct variable to a fresh new constant following the query containment tests described in [33].

THEOREM 1. *Let $\Sigma$ be a set of tgds and egds with terminating chase. A conjunctive query (CQ) $q$ is recoverable wrt. $\Sigma$, a projection $\mathscr{M}_p$ and the query class CQ, if and only if $q$ passes the recoverability test.*

```
recoverable(q, Σ, Σ_p)
Input: A conjunctive query q, a set of dependencies Σ, and a projection Σ_p
Output: Returns true if q is recoverable after projection

01  If chasing $q$ against $\Sigma$ fails, return true;
02  else let $q' = chase_\Sigma(q)$.
03  Freeze $q'$ as a database $d_1$
04  Perform the following:
05      $d_2 = chase_{\Sigma_p}(d_1)$
06      Reverse the arrows in $\Sigma_p$ resulting in $\Sigma_p^{-1}$;
07      $d_3 = chase_{\Sigma_p^{-1}}(d_2)$
08  $d_4 = chase_\Sigma(d_3)$
09  If the frozen head of $q'$ is contained in $q'(d_4)$ return true;
10  else return false
```

**Figure 2: Algorithm for Testing CQ Recoverability**

PROOF SKETCH. The case when the chase on line 1 fails is straightforward. We prove in the following the case when the chase succeeds.

$\Rightarrow$: Assume there exists a query $q_1$ over the projected schema that recovers $q$. Let $q_2$ be the unfolding of $q_1$ against the projection, we have $q_2 \equiv_\Sigma q'$. We also know $q_2(d_1) \subseteq q_2(d_4)$, since $q_2$ does not make use of any source data not exported by the projection. Therefore, $q'.frozenHead \in q'(d_1) = q_2(d_1) \subseteq q_2(d_4) = q'(d_4)$, i.e., the test succeeds.

$\Leftarrow$: We define a melting transformation, denoted by $f^{-1}$, which sends each frozen constant to its original variable but keeps variables introduced during chase unchanged. In this way, a frozen database is transformed into a CQ. We claim that $f^{-1}(d_2) \equiv_\Sigma q$ is a recovery query. □

EXAMPLE 8. *We perform the recoverability test over Example 7. Chasing against $\Sigma$ leads to the query $q' = \pi_{cname}(company \bowtie org)$. After freezing we get: $d_1 = \{company(id_0, name_0, city_0), org(id_0, name_0)\}$. After the expanding following projection, we get $d_3 = \{org(id_0, name_0), company(id_0, Name_1, city_0)\}$ with $Name_1$ being a new variable. Chasing $d_3$ against $\Sigma$ gives $d_4 = \{org(id_0, name_0), company(id_0, name_0, city_0)\}$. As $name_0 \in q'(d_4)$, the original query passes the recoverability test.*

The recoverability test can be extended to the language of union of CQs (UCQs):

PROPOSITION 3. *A UCQ, with no component CQ contained in another, is recoverable wrt. UCQ if and only if each component CQ is recoverable wrt. CQ. In particular, a CQ is recoverable wrt. UCQ if and only if it is recoverable wrt. CQ.*

THEOREM 2. *Let $\Sigma$ be a set of weakly acyclic tgds and egds, and q be a CQ, the recoverability is NP-complete wrt. $|q|$. Moreover, when each dependency in $\Sigma$ has a bounded length, and q is either of bounded length or project-free, then the recoverability test is in PTIME wrt. the size of $\Sigma$, q, and S.*

PROOF SKETCH. Since the data complexity of chasing against weakly acyclic tgds is PTIME [19], the database $d_4$ is of polynomial size. It is straightforward that embedding the body of $q'$ via a homomorphism to $d_4$ is in NP. The NP-hardness is due to a reduction from rewriting CQ using conjunctive views. When each dependency has a bounded length, the chase result is again of polynomial size. Hence the final database is of polynomial size. When $q$ is of bounded length or project-free, testing embedding can be done in PTIME. □

```
reducible(M_p, M_w)
Input: a projection $\mathscr{M}_p = (G, G', \Sigma_p)$ to be tested and a mapping $\mathscr{M}_w = (G, S, \Sigma_w)$. $\Sigma_G$ denotes the dependencies in $G$.
Output: returns true if the projection is reducible.

01  for each $r \in S$
02      $q_r$ = unfolding of $r$ against $\Sigma_w$
03  endfor
04  return $\bigwedge_{r \in S} recoverable(q_r, \Sigma_G, \Sigma_p)$
```

**Figure 3: Reducibility Test**

### 4.2.2 Projection Reducibility

Given a complete initial output mapping, a projection is *reducible* with respect to a mapping language $L$, if the induced mapping is complete with respect to $L$. The existence of a reducible projection suggests that the mediated schema in the original output mapping has redundant columns. Therefore, testing projection reducibility is in fact testing schema redundancy.

The following theorem states that projection reducibility can be tested by query recoverability tests, which we have already developed in Section 4.2.1.

THEOREM 3. *Let $\mathscr{M} = (S, G, \Sigma_{sg} \cup \Sigma_g)$ be the canonical output mapping with a witness mapping $\mathscr{M}_w$. The following two statements are equivalent:*

1. *A projection $\mathscr{M}_p$ of G is reducible wrt. full s-t tgds;*
2. *For each relation $r \in S$, unfolding of the identity query against $\mathscr{M}_w$ is recoverable wrt. $\Sigma_g$, $\mathscr{M}_p$ and UCQs.*

PROOF SKETCH. It is obvious that if all identity queries are recoverable, we just take the union of the recoverable queries of all source relations to get a witness mapping for the induced mapping system. The other direction holds because each valid instance of the target schema of the canonical mediated schema is a ground core [20] of some source instance, and in this case retaining certain answers of CQs implies the identity queries are recoverable. □

Taking into consideration that an identity mapping is a witness mapping for the canonical output mapping and proposition 3, we are now able to develop a procedure for testing projection reducibility over the canonical mediated schema, which is described in Fig. 3.

LEMMA 2. *For inputs with terminating chase, a projection of the canonical mediated schema is reducible wrt. full s-t tgds if and only if it passes the reducibility test.*

EXAMPLE 9. *Consider the mediated schema G and the output mapping $\mathscr{M}_o$ as in Ex. 4. We test whether the projection $\mathscr{M}_p$ is reducible. Recall that the witness mapping for the canonical output mapping system is in a simple form, i.e., a set of copy tgds. The corresponding unfolding of the identity query of each source predicate $r(\vec{X})$ will be $r'(\vec{X})$. Since only $grant'$ is affected in the projection $\mathscr{M}_p$, all other unfoldings are trivially recoverable. Therefore, in order to test whether $\mathscr{M}_p$ is reducible, we only need to test whether the query $q(G, P, A, S) \leftarrow grant'(G, P, A, S)$ is recoverable. The recoverability test returns false, which is in line with our previous discussion.*

```
$enumMaxProjection(I_l, C_l, l, MP, \mathscr{M}_w)$
Input: a projection set of columns $I_l$, a set $C_l$ of possible columns to be further projected, the recursion level $l$, the set $MP$ of maximal projections computed already, a set of dependencies $\Sigma$, and a witness mapping $\mathscr{M}_w$.
Output: a set of maximal sets of projections $MP$.

01 for each $x \in C_l$
02     $I_{l+1} = I_l \cup \{x\}$
03     $P_{l+1} = \{y \in C_l | y > x\}$
04     if $\exists I' \in MP$ such that $I_{l+1} \cup P_{l+1} \subseteq I'$
05         return
06     endif
07     $C_{l+1} = \emptyset$
08     for each $y \in P_{l+1}$
09         if $reducible(I_{l+1} \cup \{y\}, \mathscr{M}_w)$
10             $C_{l+1} = C_{l+1} \cup \{y\}$
11         endif
12     endfor
13     if $C_{l+1} = \emptyset \wedge \neg\exists I' \in MP$ such that $I_{l+1} \subseteq I'$
14         $MP = MP \cup \{I_{l+1}\}$
15     else
16         $enumMaxProjection(I_{l+1}, C_{l+1}, l+1, \text{MP}, \mathscr{M}_w)$
17     endif
18 endfor
```

**Figure 4: Maximal Projection Enumeration Algorithm**

## 4.3 Schema Minimization

With the schema redundancy test at hand, we are now able to search for minimal mediated schemas. A naive way of finding minimal subschemas of the canonical mediated schema that retain completeness is to enumerate all possible projections and test for reducibility. If a projection is maximal in terms of set inclusion relationship, namely any superset will not retain completeness, the corresponding induced output mapping is a desired minimal mediated schema. Obviously, the above enumeration procedure is exponential. We present a more efficient enumeration algorithm by making use of the following property: if a projection is reducible, then projecting out any subset also results in a reducible subschema. This is known as the A-priori property [2]. We use here a variant of the depth-first GenMax algorithm [22], which makes use of both superset pruning (as in the original A-priori) and subset pruning. Fig. 4 depicts the procedure to compute the maximal projections. As input it receives a set $I_l$ of columns projected out, a set of candidate columns $C_l$ that possibly can be projected out additionally, the recursion level $l$, an initially empty set of result projections, and a witness mapping $\mathscr{M}_w$. After termination $MP$ contains all the maximal projections representing minimal induced mappings. For brevity, we denote in the algorithms in Fig. 4 and Fig. 5 a projection by the set of positions to be projected out instead of explicitly constructing a projection mapping.

EXAMPLE 10. *Consider the full running example specified in Figure 1, with all integrity constraints and mappings ($M_1$ to $M_4$). The merging algorithm reveals four different ways to reduce redundancy in the mediated schema:*

1. *remove the whole* project *relation and* company.cname*;*
2. *remove the relations* project *and* org*;*
3. *remove* company.cname *and* financial.year*;*
4. *remove the relation* org *and* financial.year.

```
merge(S, Σ)
Input: joint source schema $S$, input dependencies $\Sigma$
Output: a series of output mappings $Result = \{\mathscr{M}\}$

01 compute the canonical output mapping $\mathscr{M}_o = (S, G, \Sigma_{copy} \cup \Sigma_G)$
02 construct the witness mapping $\mathscr{M}_w$ for $\mathscr{M}_o$
03 $C = \emptyset$
04 let $C'$ be the set of all columns in any relations in $G$
05 for each $c \in C'$
06     if $reducible(\{c\}, \mathscr{M}_w)$
07         $C = C \cup \{c\}$
08     endif
09 endfor
10 $MP = \emptyset$
11 $enumMaxProjections(\emptyset, C, 0, MP, \mathscr{M}_w)$
12 if $MP = \emptyset$ return $\{\mathscr{M}_o\}$
13 for each $P \in MP$
14     construct subschema $G'$ defined by P
15     construction projection mapping $\mathscr{M}_p$ determined by $P$
16     $\mathscr{M} = \mathscr{M}_o \circ \mathscr{M}_p$
17     $Result = Result \cup \{\mathscr{M}\}$
18 endfor
19 return Result
```

**Figure 5: The complete merging algorithm**

## 4.4 The Full Picture

The complete merging algorithm is depicted in Figure 5. The algorithm starts by creating the canonical mediated schema, taking into consideration all the egds and tgds in the input mapping and integrity constraints incorporated in the source schemas, if any. An initialization step finds all the projections of size one that are reducible. The initial candidate set is then fed to the enumeration algorithm. The reducibility test is employed by the enumeration algorithm to detect redundancy in the canonical mediated schema. Each maximal projection determining a minimal output mapping is added to the output.

THEOREM 4. *Let $\Sigma$ be a set of tgds and egds over S with terminating chase. The algorithm merge$(S, \Sigma)$ produces all and only the subschemas of the canonical mediated schema with an output mapping that is complete (with respect to full s-t tgds), integrated, and minimal.*

PROOF. The result simply follows the correctness of the GenMax algorithm [22], Lemma 1, Lemma 2, and Proposition 1. □

**Language for Output Mappings** The output mapping $\mathscr{M}$ generated by the algorithm is the composition of $\mathscr{M}_o$, which is a union of a set of full s-t tgds and a set of target dependencies, and $\mathscr{M}_p$, which is a special form of full s-t tgds. A recent result by Arenas et al. [3] shows that composition of two mappings specified by s-t tgds with target dependencies is able to be expressed using source-to-target second order dependencies (s-t SO dependencies). Therefore, s-t SO dependencies can be used as the syntax for the output mapping. An alternative is to use predicates in the canonical mediated schema $G_0$ as helper predicates and express the output mapping as a set of egds and tgds over $(S, G_0, G)$. The latter is our current implementation for the output mapping language. Implementation of s-t SO dependencies is in our research agenda.

**Processing Queries Over Mediated Schema** Given a user query against the mediated schema, there are two strategies for query processing. The first way is query answering using a universal solution [19], while the second is query rewriting. When the input data dependencies admit a terminating chase, the output mapping generated by our approach is guaranteed to admit also a terminating chase. Therefore, query answering can be performed by chasing the source instances against the output mapping and then perform query evaluation over the materialized instance of the mediated schema. However, the expressive mapping language we allow imposes challenges on query rewriting, since the output mapping may involve recursion of relations. We detail here an algorithm which is able to rewrite a conjunctive query into a Datalog program when the input mapping consists of weakly acyclic tgds [19], which is an extension of the inverse rule algorithm for query rewriting in Local-As-View systems [16]. The rewriting algorithm proceeds in three stages. In the first stage, bottom-up generation of functional patterns of predicates are performed until a fixpoint is reached. This stage can be shared by rewriting of different queries against a given mediated schema. In the second stage, for a given user query, we check the reachability of patterned predicates backward from the given query. The third stage takes as input the reachable patterned rules obtained in the previous stage and employs the predicate-split [16] technique to produce a function-free Datalog program.

## 5. EXPERIMENTS

The merging algorithm is implemented using Java SE 6 and SWI-Prolog 5.8.0. A parallel chase [13] is implemented in Prolog for reducibility tests. A-priori enumeration is implemented in java. The experiments have been carried out on a 2GHz dual core computer. The maximal heap size is set to 512M, while the initial heap size is 40M. Disk I/O costs are excluded from profiling, while communication costs between Java and Prolog are included. Two sets of experiments are carried out. The first are performed over real data sets to evaluate effectiveness for practical merging scenarios. The second are carried out over a workload of various degrees of complexities to demonstrate the scalability.

### 5.1 Effectiveness over Real World Data Sets

Experiments are carried out over the real world data sets from Illinois Semantic Integration Archive (`http://pages.cs.wisc.edu/~anhai/wisc-si-archive/`). Three out of five data sets are used: *Courses*, *Real Estate II*, and *Inventory*. The remaining data sets are *Faculty* and *Real Estate I* . The former has an identical schema for all data sources and hence is of little interest for schema merging. The latter is left out because it is a variant of *Real Estate II*, with less complicated mappings.

**Expressiveness of Mapping Language** Complex relationships involving joins of relations arise in the data sets, which confirms the necessity of tgds as the mapping language. Furthermore, we see it is crucial to be able to specify integrity constraints over source schemas. Without presence of keys in the source, e.g., *house* in *RealEstate II*, no attribute is projectable, even many attributes are asserted to be equivalent. This is in line with completeness: when there is no functional dependency, a tuple is only retrievable when all components are kept. The *Courses* data set demonstrates the strength of our approach to perform n-ary merge. The *Courses* data set consists of five heterogeneous sources for courses. Instead of consecutive binary merging, we utilize tgds from four binary mappings among the five data sources and perform only one merging. This is particularly suitable in scenarios where only few fixed mappings are available, such as in a P2P setting. Value conversion functions and arithmetic expressions arising in the data sets are not directly expressible in tgds. We handle the problem by using skolem functions as in SO tgds [21]. Non-invertible functions are expressed as skolem terms in the head of tgds, while invertible functions are handled using helper predicate and rules to avoid recursive nesting of skolem functions during chase. An example is the concatenation of names, for which a ternary predicate $concat(FirstName, LastName, FullNmae)$ is introduced together with two rules: $concat(split\_first(N), split\_last(N), N) \leftarrow N \backslash= concat2(\_,\_), !.$ and $concat(F, L, concat2(F, L)) \leftarrow true$. We conclude that the language of tgds as in our approach is rich enough and necessary to capture real world relationships among data sources.

**Comparison to Manually Created Schemas** Our merging algorithm removed a large number of redundant attributes from the mediated schema for all three data sets: 31 over 70 (Courses), 16 over 57 (RealEstate II), 32 over 89 (Inventory). There is a manually created mediated schema available for *Course*. Our mediated schema differs from the referential schema in several aspects. First, some source attributes occurring in only one source are left out in the referential schema while our algorithm retains them due to the requirement of completeness. Second, our generated schema is more normalized than the referential schema, which is probably due to the fact that we follow a reduction based approach. Third, since we perform multi-way merging of five course schemas using only four binary mappings among them, some correspondences of attributes are not transitively captured in the mapping simply because the intermediate schema in-between does not have an equivalent attribute. An example is *course_rice* and *course_wsu* have no direct mapping and are mapped independently to *course_reed*. *course_reed* does not have an attribute for *comment*. Therefore, comments from the two schemas are not revealed to be equivalent. The conclusion is our approach has successfully reduced schema level redundancy in real world scenarios.

### 5.2 Scalability

**Workload Generator** A random workload generator has been implemented to provide inputs of varied complexities. The workload generator consists of three components: a schema generator, a match generator, and a mapping generator. Given a set of configurations, the schema generator generates a universal relation and a set of functional dependencies, which are then fed to a top-down decomposition based schema normalizer. The schema normalizer takes as input a universal relation and a set of functional dependencies, performs a normalization, and creates a database schema in BCNF with functional dependencies and acyclic inclusion dependencies. The match generator uniformly selects one position from each schema to form a value correspondence. Each value correspondence pair is selected independently of previously selected pairs. A cyclicity test is performed before each candidate value correspondence is admitted into a match, i.e., a set of correspondences, so that the implied mapping is weakly acyclic. Given a match, the mapping generator generates a tuple generating dependency using a simplified version of query discovery [18].

**Scalability Experiment Result** We generated 166 random inputs ranging from 10 to 200 attributes, with the size of the dependency graph (covering mapping and ICs) from 10 to 5000. The running time wrt. the size of the dependency graph is demonstrated in Figure 6. The time grows following a polynomial trend line. A close look at the generated workload reveals that most of them have only a small number of redundant attributes (165 out of the 166 inputs have less than 3 redundant attributes). Therefore, the enumeration of the candidate position sets does not take much time, which in worst case can be exponential. Therefore, the running time is determined by the reducibility test, which is in PTIME when the query recoverability test is in PTIME. This is in line with our previous analysis.

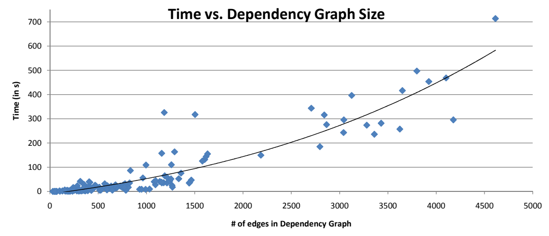

**Figure 6: Running Time vs. Size of Dependency Graph**

## 6. RELATED WORK

Schema merging has a long history in database research. We do not head for a detailed survey, but instead focus on the evolution of schema merging methodologies and compare those works to our logical approach.

Batini et al. [6] provide an early survey covering a lot of classical view/database integration approaches. As the integration tasks are usually carried out in a schema design scenario, schemas are represented in a variant of ER model or Object-Oriented model. Those approaches differ a lot on how inter-schema relationships between source schemas are represented. So called inter-schema assertions [32, 29] are a popular language specifying set-based relationships (e.g., inclusion, disjoint, and equal) between possible extensions of concepts in different schemas. Most of the approaches undergo a two phase procedure: first collapse equivalent elements in the source schemas and then resolve the conflicts arising in collapsing. Spaccapietra et al. [32] is a well known representative, which is able to handle a wide class of conflicts. Pottinger and Bernstein [27] give a categorization of various types of conflicts. A common shortcoming of this line of work is that no output mapping in the form of logical mappings (e.g., tgds) are generated, although attribute correspondences between source schemas and target schemas are an implicit result.

Schema merging using expressive logical mappings is considered to be largely unexplored [14, 8]. [10] and [11] are pioneers using logical constraints in merging. Similar to us, they consider source integrity constraints and head for a minimal mediated schema. However, their input mapping language is a special class of mappings in the form of one-to-one relation-wise implications and keys are required to be present in each implication. Our approach can be deemed as an extension of their work in the sense that we consider tuple generating dependencies which is much more expressive. Another distinction is that we consider source incompleteness, while they do not. In [28], Pottinger and Bernstein extend their early work [27] to a merging algorithm working with relational schemas and generating output mappings. Similar to us, they also head for minimal mediated schemas in data integration. Our approach differs from theirs in several ways. First, their input mapping language is a special class of GLAV mappings using conjunctive queries to specify overlap between schemas. In contrast, we consider arbitrary query containment constraints in the form of tuple generating dependencies with the only restriction that they admit a terminating chase. Second, their merging semantics is based on preserving source information and overlap. Since we do not have the concept of overlap, there is no overlap preservation in our requirements for schema merging. They assume that the extensions of the sources do not conform to any direct constraints and hence their completeness requirement is based on preserving all extensionally stored data. In contrast, we consider source incompleteness as a basic assumption and take the input mapping as expected constraints over the integrated global database. Therefore, our completeness requires preserving not only extensional data but also inferred data in the form of certain answers. Third, the queries used in their approach to witness the complete preserving of source information do not contain joins, i.e., over a single relation, while our approach uses conjunctive queries over the mediated schema to reconstruct source information, which probably leads to smaller mediated schemas. Last but not least, source integrity constraints made use of in our approach are not exploited in theirs.

As clarified in [26], view integration is a closely related but semantically different problem from data integration. View integration aims at creating a backend storage schema supporting the source schemas as views, which results in quite different requirements on the merging algorithm. Melnik [25] proposes a straightforward algorithm for view integration of logical schemas. The mediated schema is taken to be a disjoint union of the source schemas, with source dependencies and input mapping encoded as constraints. Output mappings are identity mappings copying part of the mediated schema to a corresponding source schema. Arenas et al. [4] extend the work to achieve a smaller instance for the mediated schema by adding denial constraints. The two works differ fundamentally from our work in that they head for creating a backend storage schema to support the views satisfying the input mapping. That's why their output mapping is from the mediated schema to the source schemas while we create a mapping from the sources to the mediated schema. They are more concerned with creating a smaller mediated instance of the mediated schema while the schema's size is insignificant. To the contrary, we aim at generating a minimal query interface instead of a minimal instance.

Chiticariu et al. [12] propose an interactive schema merging approach using schema matches as input. Concepts are extracted from logical schemas and each possible configuration of concept collapsing results in a plausible mediated schema. The space of plausible collapsing of concepts is then navigated by the user in an interactive manner. Since each extracted concept has a particular join path in the source schemas, two concepts and value correspondences between them comprise an implicit GLAV mapping. Following this point of view, a schema match is a representation of a collection of uncertain mapping constraints. The work is extended in [30] to generate only top-k mediated schemas, with a ranking of quality of candidate mediated schemas. A mediated schema is considered more desired if it collapses concepts with higher similarity or higher sub-element coverage. Sarma et al. [31] provide another uncertain schema merging approach using also schema matches. They represent alternative mediated schemas as a probability distribution over different clustering of attributes. A probabilistic mapping [15] is produced for each possible mediated schema. In our approach, the input mapping is constrained by logical formulas and bears no uncertainty. However, uncertainties still arise from the fact that the same piece of information can be structured in different ways. Similarly to [12], we allow multiple plausible mediated schemas as output.

We make use of the notion of witness mapping in our definition of completeness to support the recovery of certain answers. The witness mapping is from the mediated schema backward to the joint source schema. At a first glance, it is similar to the notion of mapping inverse [17] or mapping recovery [5]. However, the witness mapping is neither an inverse nor a recovery mapping of the output mapping. The composition of the output mapping and the witness mapping only need to be CQ-equivalent to the mapping $(S, S, \Sigma)$ with $\Sigma$ being the union of input tgds and source integrity constraints, and it does not need to contain the pair $(I, I)$ in the solution space,which is common for both, mapping inverse and mapping recovery, when $I$ is an incomplete joint source instance.

## 7. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel approach to n-ary schema merging for the relational model. We extend existing work by considering source integrity constraints and using a much broader class of mapping language, namely tgds with terminating chase. Under OWA, we opt for minimal mediated schemas that are complete wrt. certain answers of CQs. We have developed an algorithm producing all desired mediated schemas as results of removing redundant columns from the joint source schema. We have also described the feasibility of query processing in our framework. Finally, the evaluation has shown the applicability of our approach to real world data sets.

In future work, we will study the various requirements for schema merging under different scenarios, i.e., by investigating the semantics for other applications than virtual data integration. One possible direction is data warehousing, in which a schema is created to materialize reconciled data. Furthermore, our approach for the relational model should be extended to other popular metamodels, such as XML. Last but not least, merging schemas when the underlying data sources are not consistent is also of practical significance.

**Acknowledgements:** The authors are grateful to the anonymous referees for comments helping to improve the paper. The work is supported by the DFG Research Cluster on Ultra High-Speed Mobile Information and Communication UMIC (`www.umic.rwth-aachen.de`).

## 8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB*, pages 487–499, 1994.

[3] M. Arenas, R. Fagin, and A. Nash. Composition with target constraints. In *ICDT*, 2010.

[4] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Foundations of schema mapping management. In *PODS*, pages 227–238, 2010.

[5] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: bringing exchanged data back. In *PODS*, pages 13–22, 2008.

[6] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[7] P. A. Bernstein, A. Y. Halevy, and R. Pottinger. A vision for management of complex models. *SIGMOD Record*, 29(4):55–63, 2000.

[8] P. A. Bernstein and H. Ho. Model management and schema mappings: Theory and practice. In *Proc. VLDB*, pages 1439–1440, 2007.

[9] P. A. Bernstein and S. Melnik. Model management 2.0: Manipulating richer mappings. In *Proc. SIGMOD*, pages 1–12, Beijing, China, 2007.

[10] J. Biskup and B. Convent. A formal view integration method. In *Proc. SIGMOD*, pages 398–407, Washington, D.C., 1986.

[11] M. A. Casanova and V. M. P. Vidal. Towards a sound view integration methodology. In *PODS*, pages 36–47, Atlanta, GA, 1983. ACM.

[12] L. Chiticariu, P. G. Kolaitis, and L. Popa. Interactive generation of integrated schemas. In *Proc. SIGMOD*, pages 833–846, 2008.

[13] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *PODS*, pages 149–158, 2008.

[14] A. Doan and A. Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.

[15] X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. *VLDB J.*, 18(2):469–500, 2009.

[16] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. PODS*, pages 109–116, 1997.

[17] R. Fagin. Inverting schema mappings. In *Proc. PODS*, pages 50–59, 2006.

[18] 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: Foundations and Applications*, volume 5600 of *LNCS*, pages 198–236. Springer, 2009.

[19] R. Fagin, P. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336:89–124, 2005.

[20] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.

[21] 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.

[22] K. Gouda and M. J. Zaki. Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining & Knowledge Discovery*, 11(3):223–242, 2005.

[23] R. Hull. Relative information capacity of simple relational database schemata. *SIAM Journal of Computing*, 15(3):856–886, August 1986.

[24] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[25] S. Melnik. *Generic Model Management: Concepts and Algorithms*. PhD thesis, Universität Leipzig, 2004.

[26] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *Proc. VLDB*, pages 120–133. Morgan Kaufmann, 1993.

[27] R. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *Proc. VLDB*, pages 826–873, 2003.

[28] R. Pottinger and P. A. Bernstein. Schema merging and mapping creation for relational sources. In *Proc. EDBT*, 2008.

[29] C. Quix, D. Kensche, and X. Li. Generic schema merging. In *Proc. CAiSE'07*, volume 4495 of *LNCS*, pages 127–141, 2007.

[30] A. Radwan, L. Popa, I. R. Stanoi, and A. A. Younis. Top-k generation of integrated schemas based on directed and weighted correspondences. In *Proc. SIGMOD*, pages 641–654, 2009.

[31] A. D. Sarma, X. Dong, and A. Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *Proc. SIGMOD*, pages 861–874, 2008.

[32] S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1(1):81–126, 1992.

[33] J. D. Ullman. Information integration using logical views. In *Proc. ICDT*, pages 19–40, Delphi, Greece, 1997. Springer.