Data Sharing and Exchange: General Data-Mapping Semantics

Rana Awada(✉) and Iluju Kiringa(✉)

University of Ottawa, EECS, Ottawa, Ontario, Canada
{rawad049,iluju.kiringa}@uottawa.ca

Abstract. Traditional data sharing and exchange solved the problem of exchanging data between applications that store same information using different vocabularies. We discuss in this paper the data sharing and exchange problem between applications that store *related* data which do not necessarily possess the same meaning. We first consider this problem in settings where source instances are *complete* – that is, do not contain unknown data. Then we address more collaborative scenarios where peers can store incomplete information. We define the semantics of these settings, and we provide the data complexity for generating solutions and the minimal among those. Also, we distinguish between *sound* and *complete* certain answers as semantics for conjunctive query answering.

1 Introduction

Integrating related data from independent sources which possess differences in both structure and vocabulary, is a problem that was first addressed in data coordination (DC) settings [6]. A DC setting [6] $\mathfrak{S}$ consists of two schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, and a set of mapping tables $\{\mathcal{M}\}$. DC settings introduced so far considered mapping tables $\mathcal{M}$ with general interpretation of related data; that is, a pair (a, b) holds in $\mathcal{M}$ if a is *related* to b (where a belongs to the instance of $\mathbf{S}_1$ and b belongs to the instance of $\mathbf{S}_2$), and the *related* relationship does not possess any particular meaning. Although DC amalgamates data of different applications in a single result, it still uses in this set the different vocabularies of these applications. Such a property makes it harder in applications to analyse this data and make decisions about it. We elaborate this property in the following example.

Example 1. Let $\mathfrak{S} = (\mathbf{S}_1, \mathbf{S}_2, \mathcal{M})$ be a DC setting. Assume that schema $\mathbf{S}_1$ of university $Univ_a$ consists of the relation symbols Student(sname, sage) and Enroll(sname, cname, cgrade), and schema $\mathbf{S}_2$ of $Univ_b$ consists of the relation symbols St(sname, sage) and Take(sname, cname, cgrade). Relation *Student* (*St*) stores students' names and ages information. Relation *Enroll* (*Take*) stores the set of courses that each student completed with the final grades that he/she received in these courses. In addition, assume that $\mathbf{S}_1$ and $\mathbf{S}_2$ are connected by the mapping table $\mathcal{M}$ which contains the pairs: $\{(\text{Cid}_a,\text{Cid}_1), (\text{Cid}_a,\text{Cid}_2)\}$. Let $I_1 = \{\text{Student(Alex,18)}, \text{Enroll(Alex,Cid}_a\text{,80)}\}$ be an instance of $\mathbf{S}_1$, and I_2

S. Kozielski et al. (Eds.): BDAS 2015, CCIS 521, pp. 159–169, 2015.
DOI: 10.1007/978-3-319-18422-7_14

= {St(Ben,19), Take(Ben,Cid_1,B), Take(Ben,Cid_2,C)} be an instance of $\mathbf{S}_2$. To retrieve the list of students from both universities that completed course Cid_a or those corresponding to it according to $\mathcal{M}$, authors in [6] rewrite a query Q posed to $Univ_a$ to a query Q' using $\mathcal{M}$ then pose it to $Univ_b$. Query Q' can be: Select *sname*, *cname* From $Take$ Where *cname* = Cid_1 Or *cname* = Cid_2. The combined result of Q and Q' is {(Alex,Cid_a), (Ben,Cid_1), (Ben,Cid_2)}.

In Example 1, the combined result of Q and Q' has elements Cid_a, Cid_1, and Cid_2 which are different vocabularies. Such a property makes it hard to compare the performances of Alex and Ben in a certain course. In DC [6], one source element can be mapped in $\mathcal{M}$ to one or more target elements. Clearly, a higher number of mapped elements lead to bigger sizes of combined results, and more expensive query computations. For some queries, retrieving a portion of the result can be informative enough for users to escape the more expensive computation of the full result.Therefore, to control the size or related information returned, authers in [6] introduced the notions of complete and sound query translations that allow users compute the full set of correct answers of conjunctive queries (CQs), called *complete* answers, and a portion of those, named *sound* answers, respectively. From Example 1, query Q' is a complete query translation, while a sound query translation can be Q'' = Select *sname*, *cname* From $Take$ Where *cname* = Cid_1 (since both Cid_1 and Cid_2 are mapped to Cid_a in $\mathcal{M}$).

Our main interest in this work is to solve the heterogeneity problem in DC settings while allowing users to generate sound and complete answers of CQs. Intuitively, one possible solution is to exchange data from independent sources and store it in a separate target using a unified set of vocabularies, prior to applying queries. It turns out that data sharing and exchange (DSE) settings introduced recently in [3] provides such a functionality, and in particular, DSE solutions in this setting suits well our requirement to generate sound and complete answers. As given in [3], a DSE setting is a tuple $\mathfrak{S} = (\mathbf{S},\mathbf{T},\mathcal{M},\ \Sigma_{st})$, where $\mathbf{S}$ and $\mathbf{T}$ are source and target schemas, respectively; $\mathcal{M}$ is an st-mapping table, and Σ_{st} consists of a set of mapping *source-to-target tuple-generating* dependencies (st-tgds). DSE [3] however supports applications that refer to same elements using different sets of vocabularies; that is, they use the equivalence semantics of related data in st-mapping tables.

We discuss in this paper a DSE setting, denoted by DSE_G, to exchange *related* information between applications that possess different schemas and different domains of constants. Our main contributions are the following: (1) we provide algorithms to generate DSE solutions in LOGSPACE, (2) we provide algorithms to generate more compact universal DSE solutions, minimal universal DSE solutions, in LOGSPACE, (3) we formally define sound and complete certain answers as semantics for CQ answering and we provide algorithms to generate those, (4) finally, we address the DSE_G problem in more challenging settings where source instances contain unknown information in the form of *nulls*.

2 Preliminaries and Related Work

We start this section by defining the notations that we will use through our the paper. Then we give a brief summary of two settings defined previously in the literature to address the data exchange and data coordination problems. These are data exchange (DE) [5] and DSE [3] settings.

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity $n_i > 0$. Let D be a countably infinite domain. An *instance* I of $\mathbf{R}$ assigns to each relation R_i of $\mathbf{R}$ a finite n_i-ary relation $R_i^I \subseteq \mathsf{D}^{n_i}$. Sometimes we write $R_i(\bar{t}) \in I$ instead of $\bar{t} \in R_i^I$, and call $R_i(\bar{t})$ a *fact* of I. The *domain* $dom(I)$ of instance I is the set of all elements that occur in any of the relations R_i^I. We often define instances by simply listing the facts that belong to them.

Data Exchange Settings. Data exchange [5] is the problem of extracting an instance over a source schema $\mathbf{S}$ and transform it to confirm to an independent target schema $\mathbf{T}$. More formally, a DE setting is a tuple $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is an independent target schema, Σ_{st} is a set of st-tgds which are FO sentences of the form $\forall\bar{x}\forall\bar{y}\forall\bar{z}\,(\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{w}\,\psi(\bar{z},\bar{w}))$, where $\phi(\bar{x},\bar{y})$ and $\psi(\bar{z},\bar{w})$ are conjunctions of relational atoms over $\mathbf{S}$ and $\mathbf{T}$, respectively. It was customary in DE to exchange data based on st-tgds that consists of positive predicates and equality formulas, until authors in [4] introduced a solution that solved the problem of exchanging data using st-tgds which contain negated predicates and/or inequality formulas. In their solution, they extended the source schema $\mathbf{S}$ to $\hat{\mathbf{S}} := \mathbf{S} \cup \{\hat{R} : \neg R \in \Sigma_{st}\} \cup \{N\}$. They also re-constructed Σ_{st} to $\hat{\Sigma}_{st}$ by replacing each negated literal $\neg R(\bar{x})$ with $\hat{R}(\bar{x})$, and inequality $x \neq y$ over $\mathbf{S}$ with and $N(x, y)$ over $\hat{\mathbf{S}}$, respectively. Each table $\hat{R}$ is populated with the evaluation of $\neg R(\bar{x})$ on the instance I of $\mathbf{S}$, and $N(x, y)$ contains all the pairs of elements (a, b) in $dom(I)$ such that $a \neq b$.

An instance J of a target schema $\mathbf{T}$ is said to be a *solution* for a source instance I under $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, if the instance (I, J) of $\mathbf{S} \cup \mathbf{T}$ satisfies Σ_{st}. The DE literature identified a class of "good" solutions, called the *universal* solutions, that in a precise way represent all other solutions. To define those, authors in [5] used the notion of homomorphism between instances. Let K_1 and K_2 be instances of the same schema $\mathbf{R}$. A *homomorphism* h from K_1 to K_2 is a function $h : dom(K_1) \rightarrow dom(K_2)$ such that: (1) $h(c) = c$ for every $c \in \mathsf{Const} \cap dom(K_1)$, and (2) for every $R \in \mathbf{R}$ and tuple $\bar{a} = (a_1, \ldots, a_k) \in R^{K_1}$, it holds that $h(\bar{a}) = (h(a_1), \ldots, h(a_k)) \in R^{K_2}$. Let $\mathfrak{S}$ be a DE setting, I a source instance and J a solution under $\mathfrak{S}$. Then J is a *universal* solution under $\mathfrak{S}$, if for every solution J' for I under $\mathfrak{S}$, there exists a homomorphism from J to J'.

Data Sharing and Exchange Settings. DSE [3] exchanges data between applications that refer to same objects using different instance values. Formally, a DSE setting is a tuple $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$, where: (1) $\mathbf{S}$ and $\mathbf{T}$ are a source and a target schema, respectively; (2) $\mathcal{M}$ is a binary relation symbol that appears neither in $\mathbf{S}$ nor in $\mathbf{T}$, and that is called a *source-to-target (st-) mapping* and (3) Σ_{st} consists of a set of *mapping* st-tgds, which are FO sentences of the form $\forall\bar{x}\forall\bar{y}\forall\bar{z}\,(\phi(\bar{x},\bar{y}) \wedge \mu(\bar{x},\bar{z}) \rightarrow \exists\bar{w}\,\psi(\bar{z},\bar{w}))$, where (i) $\phi(\bar{x},\bar{y})$ and $\psi(\bar{z},\bar{w})$ are

conjunctions of relational atoms over $\mathbf{S}$ and $\mathbf{T}$, resp., (ii) $\mu(\bar{x}, \bar{z})$ is a conjunction of atomic formulas that only use the relation symbol $\mathcal{M}$.

While data mappings possessed no particular meaning in DC settings [6], authors in [3] adopted the *equivalence* semantics in st-mapping tables; that is two elements a and b are mapped in an st-mapping table $\mathcal{M}$ only if they possess the same meaning. Apparently, and as discussed in [3], such equivalence interpretation of related data can entail new equivalent elements in $\mathcal{M}$. For example, if an element a is related to elements a' and a'' in $\mathcal{M}$ and element b is related to elements a' and b' in $\mathcal{M}$, then as a consequence elements a'' and b' are considered equivalent according to the semantics of $\mathcal{M}$ and a can be mapped to b' in $\mathcal{M}$. This equivalence property in st-mapping tables made the knowledge exchange framework [2] be a natural representation for DSE [3]. A knowledge base (KB) over a schema $\mathbf{R}$ is a pair (K, Σ), where K is an instance of $\mathbf{R}$ (the explicit data) and Σ is a set of logical sentences over $\mathbf{R}$ (the implicit data).

In DSE [3], source and target instances store explicit data accompanied with implicit information in the form of full tgds[1] – denoted by Σ_s and Σ_t respectively– that complete the source KB, the st-mapping table, and the target KB with the entailed information. They also defined a class of "good" solutions – they called it universal DSE solutions – that could be generated in LOGSPACE.

Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting [3], I a source instance, $\mathcal{M}$ an st-mapping table, J a target instance. Denote by $K_{\mathbf{R}'}$ the restriction of instance K to a subset $\mathbf{R}'$ of its schema $\mathbf{R}$. Recall that Σ_s, Σ_t are the source and target completions of $\mathfrak{S}$, respectively. Then, J is a DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, if for every $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t)$[2] there is $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}), \Sigma_s)$ such that the following hold: (a) $K'_{\mathcal{M}} \subseteq K_{\mathcal{M}}$, and (b) $K_{\mathbf{T}} \vDash ((K'_{\mathbf{S}} \cup \{K'_{\mathcal{M}}\}), \Sigma_{st})$ under $\mathfrak{S}$. In addition, J is a *universal* DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, if J is a DSE solution, and for every $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}, \Sigma_s)$ there is $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}, \Sigma_t)$ such that (a) $K_{\mathcal{M}} \subseteq K'_{\mathcal{M}}$, and (b) $K_{\mathbf{T}} \vDash ((K'_{\mathbf{S}} \cup \{K'_{\mathcal{M}}\}), \Sigma_{st})$ under $\mathfrak{S}$.

3 General Data Sharing and Exchange

As before, instances of $\mathbf{S}$ (resp. $\mathbf{T}$) are called source (resp. target) instances[3]. Instances of $\mathcal{M}$ are called st-mapping tables[4]. Also, as the case in [3], we assume the existence of two (not necessarily disjoint) countably infinite sets of constants $\mathsf{Const}^{\mathbf{S}}$ and $\mathsf{Const}^{\mathbf{T}}$, that denote the set of source and target constants, respectively. We also assume the existence of a countably infinite set Var of labeled nulls (that is disjoint from both $\mathsf{Const}^{\mathbf{S}}$ and $\mathsf{Const}^{\mathbf{T}}$). For the time being we assume that $dom(I) \in \mathsf{Const}^{\mathbf{S}}$. We will drop this assumption later in section 4. Also, as the case in [3], st-mapping tables are over $(\mathsf{Const}^{\mathbf{S}}, \mathsf{Const}^{\mathbf{T}})$.

[1] A full tgd is a tgd that does not contain existentially quantified variables.

[2] $\mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t)$ is defined as the set of instances of $(\mathbf{T} \cup \{\mathcal{M}\})$ that contain the explicit data in $(J \cup \{\mathcal{M}\})$ and satisfy the implicit data in Σ_t.

[3] We denote source instances by $I, I', I_1, \ldots$ and target instances by $J, J', J_1, \ldots$.

[4] We slightly abuse notations and denote both st-mapping relations and st-mapping tables by $\mathcal{M}$.

Formally, a DSE$_G$ setting $\mathfrak{S}$ is a setting where the definition of a DSE setting introduced in [3] applies to $\mathfrak{S}$. We consider in DSE$_G$ source KBs of the form $((I \cup \{\mathcal{M}\}), \emptyset)$, which correspond to data in the source instance I and the st-mapping table $\mathcal{M}$. On the other hand, the target KBs are of the form $((J \cup \{\mathcal{M}\}), \Sigma_t)$, where J contains a portion of the exchanged data, and Σ_t contains a set of FO sentences, of type full tgds, over a schema that includes $\mathbf{T}$ and $\mathcal{M}$.

We assume in a DSE$_G$ setting that the st-mapping table $\mathcal{M}$ possesses the following particular characteristic: for each value $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$, there exists an element $a' \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ such that $\mathcal{M}(a, a')$ holds, and for no $b \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$ – different than a – $\mathcal{M}(b, a')$ holds. We say a' uniquely identifies a in $\mathcal{M}$. More formally, we assume $\mathcal{M}$ satisfies the following constraint: $\forall x \exists y \forall z(\mathcal{M}(x, y) \wedge \mathcal{M}(z, y) \rightarrow x = z)$. To be used later, we store in a fresh table C, defined in both schemas $\mathbf{S}$ and $\mathbf{T}$, the set of values in $(dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ that uniquely identify source elements mapped in $\mathcal{M}$. So, following Example 1, and assuming that $\mathcal{M}$ includes the pair $\{(\text{Cid}_b,\text{Cid}_2), (\text{Cid}_b,\text{Cid}_3)\}$, then Cid_1 uniquely identifies Cid_a and Cid_3 uniquely identifies Cid_b, and $C = \{\text{Cid}_1, \text{Cid}_3\}$.

We adopt (universal) DSE solutions introduced in [3] with the restriction of $\Sigma_s = \emptyset$ to define the semantics of DSE$_G$. We illustrate DSE solutions in DSE$_G$ settings in the following example.

Example 2. (Example 1 cont.) In reference to the DC setting $\mathfrak{S}$ given in Example 1, let I_1 be the source instance and $\mathcal{M}$ the st-mapping table in $\mathfrak{S}$. Also, assume that the set of mapping st-tgds Σ_{st} is the following:
(*a*) $Student(x, y) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y') \rightarrow St(x', y')$.
(*b*) $Enroll(x, w, u) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(w, w') \wedge \mathcal{M}(u, u') \rightarrow Take(x', w', u')$.

Then a possible DSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ would be $J = \{St(Alex, 18), Take(Alex, Cid_1, B), Take(Alex, Cid_2, B)\}$.

Clearly, C is the sole set of target values in a DSE$_G$ instance that correctly captures the set of source values exchanged to a target instance. Therefore, we use C as a fundamental part of the FO sentences – the implicit data – in Σ_t. The set of FO sentences d over a schema that includes $\mathcal{M}$, C, and a fresh table C', which specify the target elements in $(dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ that are in C, are the following: (1) $\forall x \forall y \forall z(\mathcal{M}(x, y) \wedge \mathcal{M}(z, y) \wedge x \neq z \rightarrow C'(y))$, (2) $\forall x \forall y(\mathcal{M}(x, y) \wedge \neg C'(y) \rightarrow C(y))$.

Authors in [4] addressed the problem of chasing dependencies which contain negated predicates and inequality formulas. We adopt this solution and apply it to the set of rules d to populate the table C. We prove the following result:

Theorem 1. *Let $\mathfrak{S}$ be a fixed DSE$_G$ setting and $\mathcal{M}$ be an st-mapping table. Then generating C is in* LOGSPACE.

We define below a target completion program as a set of full tgds, denoted Σ_t, over a schema that includes $\mathbf{T}$ and $\mathcal{M}$, such that applying Σ_t to a universal DSE solution J generates a universal DSE solution J' for I and $\mathcal{M}$ under $\mathfrak{S}$ which contains the complete set of exchanged data; that is $((I \cup \{\mathcal{M}\}), J') \vDash \Sigma_{st}$.

Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_G$ setting. Σ_t, is the following set of FO sentences over $\mathbf{T} \cup \{\mathcal{M}, \text{REL}\}$, and REL is a fresh binary table:

1. For each $T \in \mathbf{T} \cup \{\mathcal{M}\}$ of arity n and $1 \leq i \leq n$:
   $\forall x_1 \cdots \forall x_n (T(x_1, \ldots, x_i, \ldots, x_n) \rightarrow \textsc{Rel}(x_i, x_i))$.
2. $\forall x \forall y \forall z (\mathcal{M}(z, x) \wedge \mathcal{M}(z, y) \wedge C(x) \rightarrow \textsc{Rel}(x, y))$.
3. For each $T \in \mathbf{T}$ of arity n:
   $\forall x_1, y_1 \cdots \forall x_n, y_n \, (T(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \textsc{Rel}(x_i, \, y_i) \rightarrow T(y_1, \ldots, y_n))$.

The first rule defines the reflexive relation Rel on the domain of the target instance. The second rule captures the target elements that are related to the same source value in the st-mapping table $\mathcal{M}$. The last rule allows to complete the symbols of $\mathbf{T}$, by adding elements declared to be related in Rel.

Intuitively, applying a procedure (based on the *chase* [4]) to the instance $(I \cup \{\mathcal{M}\})$, generates a universal DSE solution in DSE_G. Thus, since the chase runs in Logspace [1] and following Theorem 1, we can conclude that there exists a Logspace algorithm that generates DSE solutions in fixed DSE_G settings.

Most compact solutions in a DSE_G setting can be identified by the class of *minimal* universal DSE (MUDSE) solutions introduced in [3]. A MUDSE solution J for a source instance I and an st-mapping table $\mathcal{M}$ in DSE_G is such that (1) there exists no proper subset J' of J where J' is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, and (2) there exists no universal DSE solution J' such that $(dom(J') \cap \mathsf{Const}^{\mathbf{T}})$ is properly contained in $(dom(J) \cap \mathsf{Const}^{T})$.

Example 3. (Example 2 cont.) In reference to Example 2, a possible MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ would be $J = \{St(Alex, 18), Take(Alex, Cid_1, B)\}$.

We provide a procedure $\texttt{CompMUDSE}_{\mathsf{G}}\texttt{sol}_{\mathfrak{S}}$, a variant of $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ [3], that given an instance I and an st-mapping table $\mathcal{M}$ in a DSE_G setting $\mathfrak{S}$, generates a MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.

$\texttt{CompMUDS}_{\mathsf{G}}\texttt{sol}_{\mathfrak{S}}$:
Input: A source instance I, an st-mapping table $\mathcal{M}$, and a set Σ_{st} of st-tgds.
Output: A Canonical MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.

1. Populate in the table C elements from $\mathcal{M}$ using the set of FO sentences d.
2. Apply a procedure (based on the *chase* [4]) to the instance $(I \cup \{\mathcal{M}\})$, and generate a target instance J.
3. Compute a set of classes $\{C_1, \ldots, C_m\}$ over $dom(C)$ such that c_1 and c_2 exist in C_i if there exists a constant a such that $\mathcal{M}(a, c_1)$ and $\mathcal{M}(a, c_2)$ hold.
4. Choose a set of witnesses $\{w_1, \ldots, w_m\}$ such that $w_i \in C_i$, for $1 \leq i \leq m$.
5. Compute from J the instance $J' := \texttt{replace}(J, w_1, \ldots, w_m)$ by replacing each occurrence of target constant $c \in C_i \cap dom(J)$ $(1 \leq i \leq m)$ with $w_i \in C_i$.
6. Apply a procedure (based on the core [5]) to the target instance J' and generate the target instance J_1 that is the core of J'.

We prove the correctness of $\texttt{CompMUDSE}_{\mathsf{G}}\texttt{sol}_{\mathfrak{S}}$ and that it runs in Logspace in the following result:

Theorem 2. *Let $\mathfrak{S}$ be a fixed DSE_G setting, I a source instance, and $\mathcal{M}$ an st-mapping table. Let J^* be an arbitrary result for $\texttt{CompMUDSE}_{\mathsf{G}}\texttt{sol}_{\mathfrak{S}}$. Then, J^* is a MUDSE for I and $\mathcal{M}$ under $\mathfrak{S}$. Also, $\texttt{CompMUDSE}_{\mathsf{G}}\texttt{sol}_{\mathfrak{S}}$ runs in* Logspace.

4 DSE_G and Incomplete Source Data

We discuss in this section DSE_G in collaborative settings in which several sources interact, we denote it by DSE_I to avoid any confusion. In such scenarios, it is most likely that data in a source instance is obtained by an earlier exchange from a different source, and hence may contain null values. The most intuitive tables with null values are called naïve tables. Arenas et al. addressed in [2] the DE problem in such collaborative settings. They proved that target instances with positive conditional tables, named also as *pc-tables* [5], are the best tool to represent source instances that contain naïve tables and/or pc-tables.

Clearly, DSE_I combines both the KB exchange and the DE with incomplete information [2] concepts, and hence require applying the techniques from both settings. Authors in [2] defined a variant *chase* algorithm which generates in fixed DE settings universal solutions with pc-tables in PTIME. We combine this fact with the result given in Theorem 1 to deduce that there exists a PTIME algorithm that generates a universal DSE solution (J, Σ_t) for a source knowledge base $(I, \emptyset)$ in a fixed DSE_I setting. On the other hand, we show in the following Theorem that checking whether a given pc-table target instance is a universal DSE solution in a fixed DSE_I setting is of a higher complexity.

Theorem 3. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE_I setting where Σ_{st} is a set of fixed st-tgds, I a pc-table over $\mathbf{S}$, and J a pc-table over $\mathbf{T}$. Also let the set of implicit data Σ_t be a fixed set of full tgds. Then, checking if J is a universal DSE solution for I is DP_2^P-complete.*

The class of MUDSE solutions discussed so far in this work and in DSE settings [3], contained solely of naïve tables. We prove in the following result that the best tool to represent the most compact DSE solutions for target instances with naïve tables in DSE_I is indeed a naïve table. That is, there does not exist a DSE solution with pc-tables that is more compact under $\mathfrak{S}$.

Theorem 4. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE_I setting, I a source instance, $\mathcal{M}$ an st-mapping table, and J a MUDSE solution (with naïve tables) for I and $\mathcal{M}$ under $\mathfrak{S}$. Then there does not exist a solution J' under $\mathfrak{S}$ such that J' contains pc-tables, $\mid J' \mid$ [6] $< \mid J \mid$, and $Rep(J') = Rep(J)$* [7].

Given a DSE_I setting $\mathfrak{S}$ and a DSE solution J with pc-tables. At first, one can think that generating the smallest subset instance of J, in the favor of generating a MUDSE solution J' of J under $\mathfrak{S}$, is possible by applying the Greedy algorithm introduced in ordinary DE settings [5]. Briefly, a Greedy algorithm

[5] A pc-table is a naïve table extended with a *local* condition for each tuple. This local condition is a positive boolean combination of formulas of the form $\perp = \perp'$ and $\perp = a$, where $\perp, \perp' \in \mathsf{Var}$ and $a \in \mathsf{Const}$.

[6] $\mid T \mid$ define the number of tuples in a table T.

[7] $Rep(R) = \{\rho(R) \mid \rho : \mathsf{Var}(R) \rightarrow \mathsf{Const}$ is a valuation for the variables in table R (in case R is a pc-table, then a fact $\rho(t) \in \rho(R)$ only if the condition ψ in t is such that $\rho(t) = \text{true}$).

R

A	B	C	$Condition$
x	y	z	$true$
1	1	1	$x = 1$
2	2	2	$x = 1$
3	3	3	$x = 1$
1	1	1	$x = 2 \wedge y = 2 \wedge z = 2$

Fig. 1. DSE solution J

R

A	B	C	$Condition$
x	y	y	$y = z$
x	x	z	$x = y$
x	a	b	$true$
c	y	d	$true$
e	f	z	$true$

Fig. 2. DSE solution J_1

R'

A	B	C	$Condition$
x	y	z	$x = y \vee y = z$
x	a	b	$true$
c	y	d	$true$
e	f	z	$true$

Fig. 3. DSE solution J'_1

initially initializes a fresh instance J' to J, then checks for each tuple $t \in J'$ whether $(J' - t) \equiv J$ holds (or in other words $Rep(J' - t) = Rep(J)$). If so, it updates J' to be $J' - t$. However, we show in Example 4 that this is not the case.

Example 4. Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_I$ setting and J be a DSE solution under $\mathfrak{S}$. Assume that J consists of the pc-table R given in Fig. 1. Notice that $[R - \{\langle 1, 1, 1 : x = 1\rangle, \langle 2, 2, 2 : x = 1\rangle\}] \equiv R$ and $[R - \{\langle 3, 3, 3 : x = 1\rangle\}] \equiv R$. However, $[R - \{\langle 1, 1, 1 : x = 1\rangle, \langle 2, 2, 2 : x = 1\rangle, \langle 3, 3, 3 : x = 1\rangle\}] \not\equiv R$. Therefore, to check whether pc-table R' is the smallest subset pc-table of R, we need to non-deterministically determine whether there exists an instance $R'' \subset R$ such that $|R''| < |R'|$ and $R'' \equiv R$.

Following the intuition in Example 4 we provide the result in Theorem 5

Theorem 5. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_I$ setting where Σ_{st} is a fixed set of st-tgds, I a source instance with pc-tables over $\mathbf{S}$ and J a DSE solution with pc-tables over $\mathbf{T}$. Also, let Σ_t be a fixed set of full tgds. To check if there exists a DSE solution J' for I and $\mathcal{M}$ under $\mathfrak{S}$ such that $J' \subset J$ is $\Pi_2^{P^{\|}}$-complete.*

In some DSE$_I$ instances $\mathfrak{S}$, MUDSE solutions are not the most compact solutions under $\mathfrak{S}$. Consider the DSE solution J_1 given in Fig. 2. We can see that the DSE solution J'_1 given in Fig. 3 is such that $J_1 \equiv J'_1$, $J'_1 \not\subset J_1$, and $|J_1| > |J'_1|$. Although pc-tables are proved in [2] to be the best tool to represent source instances with incomplete information, we show in the following proposition that for some DSE solutions with pc-tables there exist target instances with *conditional tables* (c-table) [8] that are equivalent and more compact than those.

Proposition 1. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_I$ setting, I a source instance, $\mathcal{M}$ an st-mapping table, and Σ_{st} be a set of st-tgds. Let J be a DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Then, the right tool to represent the most compact DSE solution J' of J is a c-table.*

5 Query Answering

We distinguish in DSE$_G$ between *sound* and *complete* certain answers as semantics for CQs answering. Let $\mathfrak{S}$ be a DSE$_G$ setting, I a source instance, and $\mathcal{M}$

[8] c-tables are more general pc-tables that can include inequality formulas, along with equalities, in their conditions.

an st-mapping table. A complete certain answer for a CQ Q over I and $\mathcal{M}$ under $\mathfrak{S}$, denoted by $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, corresponds to the set of tuples that belongs to the evaluation of Q over $K_{\mathbf{T}}$, for each DSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ and $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t)$. A sound certain answer, on the other hand, denoted by $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, would be a set P of tuples that belongs to the evaluation of Q over a DSE solution J such that applying a query completion program, denoted by Σ_q, in the style of the target completion program Σ_t, to P would regenerate the set $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$.

Computing the set $\mathsf{complete\text{-}certain}_{\mathfrak{S}}$ answers using MUDSE solutions proved in [3] to be less expensive in run times than when using universal DSE solutions completed with Σ_t. Therefore, we introduce in what follow the method to compute $\mathsf{complete\text{-}certain}_{\mathfrak{S}}$ answers of CQs using MUDSE solutions in DSE_G.

Let $\mathfrak{S}$ be a DSE_G setting, I a source instance, $\mathcal{M}$ an st-mapping table, and $Q(\bar{x}) = (\bar{x})\exists\bar{y}\ \phi(\bar{x}, \bar{y}) \wedge \psi(\bar{y})$ be a CQ over $\mathbf{T}$ where: $\bar{x}$ is a set of distinguished variables, $\phi(\bar{x}, \bar{y})$ is a conjunction of predicate formulas with distinct variables, and $\psi(\bar{y})$ is a conjunction of formulas of the form $y_1 = y_2$ where $y_1, y_2 \in \bar{y}$.

To compute $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we adopt a method similar to the combined approach given in description logic (DL) KB-exchange settings [7]. To do so, it first applies rules 1 and 2 in the target completion process Σ_t to populate the table Rel with elements entailed to be related by $\mathcal{M}$. Then, re-writes query $Q(x_1, \ldots, x_n)$ to query: $Q'(x_1, \ldots, x_n) = (\bar{x})\exists\bar{y}\exists\bar{w}\ \phi(\bar{x}, \bar{y}) \wedge \psi'(\bar{y})$ where $\psi'(\bar{y})$ constitutes the formula $\text{Rel}\ (y_1, y') \wedge \text{Rel}(y_2, y')$ for each formula $y_1 = y_2$ in $\psi(\bar{y})$, to generate the set of sound certain answers.

Intuitively, the set $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ cannot be simply obtained by posing Q to the MUDSE solution J, since J might be incomplete with respect to $\mathcal{M}$. Therefore, we present below two possible methods for computing those complete certain answers using J.

The first method would be to complete J with the information entailed by the target completion program Σ_t as a first step, and this is done by applying Σ_t to J (denoted as $\Sigma_t(J)$) and generate a complete target instance $\hat{J}$, then apply Q to $\hat{J}$ as a second step and discard tuples with null values.

A second method, on the other hand, leverages the approach we used to compute $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$. It first populates the table Rel and re-writes Q to a query Q' the same way we did to compute $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$. Then it completes the evaluation of Q' on J by returning the answer of $\hat{Q}'(z_1, \ldots, z_n) = Q'(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{Rel}(x_i, z_i)$. We prove the correctness of the above query re-writing methods in the below proposition.

Proposition 2. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE_G setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a MUDSE solution, and Q a fixed conjunctive query over $\mathbf{T}$. Then, $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \hat{Q}(J)$ where $\hat{Q}(z_1, \ldots, z_n) = (\bar{x})\exists\bar{y}\exists\bar{w}\ \phi(\bar{x}, \bar{y}) \wedge \text{Rel}(y_1, w_1) \wedge \text{Rel}(\ y_2, w_1) \wedge \cdots \wedge \bigwedge_{i=1}^{n} \text{Rel}(x_i, z_i), w_i \in \bar{w}$.*

Following Proposition 2, we deduce that the query re-writing method to compute $\mathsf{sound\text{-}certain}_{\mathfrak{S}}$ is correct. Also, based on the result in Theorem 2 that MUDSE solutions can be generated in Logspace in a fixed DSE_G setting, and

Table 1. List of Queries

$Q1$	*Fetch the name and age of each student enrolled in a course*
$Q2$	*Fetch the name and age of each student with the names of courses he completed*
$Q3$	*Fetch the name of each student with the name and grade of each course he finished*
$Q4$	*Fetch the list of teachers that already taught a course*
$Q5$	*Fetch the list of teachers' names and the list of courses they taught*
$Q6$	*Fetch the names of students with the names of courses they completed and the name of the teacher that taught each course*
$Q7$	*Fetch the list of pairs of students' names and ages that took the same course*

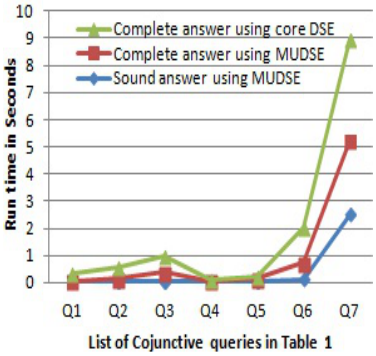

Fig. 4. Queries of table 1 run times

since checking if a fixed CQ is satisfied in a database is in LOGSPACE [2], we can deduce that computing $\mathsf{complete\text{-}certain}_{\mathfrak{S}}$ and $\mathsf{sound\text{-}certain}_{\mathfrak{S}}$ is in LOGSPACE.

6 Experiments

We conducted our experiments on a Lenovo workstation with a Dual-Core Intel(R) 1.80GHz processor, 4GB of RAM, and a 297 GB hard disk. We used PostgreSQL(v9.2) database system. We considered in our experiments the DSE_G setting of Example 1 extended with relations $Teacher\ (tname, tage)$ and $Teach\ (tname, cname)$ that specify teachers' information and the list of courses they teach, respectively. We used a DSE solution (that is a core [5]) with 55,000 tuples, where each course in the source is related to 5 courses in the target. It is clear in Fig. 1 how MUDSE solutions compute efficiently both sound and complete certain answers for CQs.

7 Concluding Remarks

We defined in this paper a DSE_G setting with general semantics of related data in an st-mapping table $\mathcal{M}$. We defined algorithms to generate DSE and MUDSE solutions. Also, we distinguished between sound and complete certain answers of CQs and we showed how to compute those. Finally, we addressed DSE_G in a setting where the source instance contain unknown information.

References

1. Arenas, M., Barceló, P., Libkin, L., Murlak, F.: Relational and xml data exchange (2010)
2. Arenas, M., Perez, J., Reutter, J.: Data exchange beyond complete data (2011)

3. Awada, R., Barceló, P., Kiringa, I.: Sharing and exchanging data (2013)
4. Deutsch, A., Nash, A., Remmel, J.: The chase revisited (2008)
5. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core (2005)
6. Kementsietsidis, A., Arenas, M., Miller, R.J.: Mapping data in peer-to-peer systems: Semantics and algorithmic issues (2003)
7. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in el using a database system (2008)