Sharing and Exchanging Data

Rana Awada[1(✉)], Pablo Barceló[2], and Iluju Kiringa[1]

[1] EECS, University of Ottawa, Ottawa, Canada
{rawad049,Iluju.kiringa}@uottawa.ca
[2] Department of Computer Science, University of Chile, Santiago, Chile
pbarcelo@dcc.uchile.cl

Abstract. Exchanging and integrating data that uses different vocabularies are two prominent problems in the database literature. These problems have been, so far, solved separately, and never been addressed together in a unified setting. In this paper, we propose a class of mappings - called *DSE*, for *data sharing and exchange* - that represents this unified setting. We introduce a DSE setting with particular interpretation of related data where ordinary data exchange or data integration cannot be applied. We define the class of *DSE* solutions in a DSE setting, that allow to store a part of explicit data and a set of inference rules used to generate the complete set of exchanged data. We identify among those a particular DSE solution with good properties; namely, one that contains a minimal amount of explicit data. Finally, we define the set of certain answers to conjunctive queries.

Keywords: Data exchange · Data coordination · Knowledge exchange

1 Introduction

Different problems of accessing and integrating data residing in independent sources have received wide attention in the literature, and different systems were introduced to solve these problems, e.g. distributed databases, federated databases, data exchange settings, and (peer-to-peer) data coordination settings.

Data exchange [7] defines the problem of moving data residing in independent applications and accessing it through a new target schema. This process of exchange only allows to move data from a source into a target that uses the same set of vocabularies, and thus, transformation occurs to the structure of the data, and not to the data itself. All data integration and coordination systems [2,14,15], on the other hand, use different query re-writing methods to allow access to data residing in independent sources, that possibly use different vocabularies, without having to exchange it and while maintaining autonomy.

We show in what follow that a collaborative process – including coordination tools for managing different vocabularies of different sources and exchange tools – would yield interoperability capabilities that are beyond the ones that can be offered today by any of the two tasks separately.

M. Hanus and R. Rocha (Eds.): KDPD 2013, LNAI 8439, pp. 17–32, 2014.
DOI: 10.1007/978-3-319-08909-6_2,

Recall that a data exchange (DE) setting [7] $\mathfrak{S}$ consists of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, and a set Σ_{st} of database dependencies – the so-called source-to-target dependencies – that describes structural changes made to data as we move it from source to target. This exchange solution supports exchanging information between two applications that refer to the same object using the same instance value. However, there exist cases where objects in the source are named and referred to differently in the target. A motivating example of such an exchange scenario is exchanging data about students applying for program transfers from one university to a different one. Indeed, different universities can offer different courses and a course in one university can possess one or more equivalent courses in the second. Given what we mentioned so far about a DE setting, we can easily deduce that DE does not support such type of exchange.

Unlike data exchange, data coordination (DC) settings [1] solved the problem of integrating information of different sources that possess different yet related domains of constants by using the *mapping table* construct [9] in their query rewriting mechanisms. A mapping table specifies for each source value the set of *related* (or *corresponding*) target values. DC settings have been studied mainly in peer-to-peer networks, where sources – called peers – possess equal capabilities and responsibilities in terms of propagating changes and retrieving related information. A DC setting $\mathfrak{S}$ consists of two schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, and a set of mapping tables $\{\mathcal{M}\}$. We give in the following example a data coordination instance that integrates data from two different universities with different domains of constants, and we show that query re-writing techniques still miss to return results that are *inferred* by certain semantics of mapping tables.

Example 1. Let $\mathfrak{S}$ be a DC setting. Suppose that $\mathbf{S}_1$ in $\mathfrak{S}$ is a schema for the University of Ottawa (UOO) and $\mathbf{S}_2$ in $\mathfrak{S}$ is the schema of the University of Carleton (UOC).

Suppose $\mathbf{S}_1$ has the relations: $Student(Sname, Sage)$, $Course(Cid, Cname, Pname)$, and $Enroll(Sname, Cid, Cgrade)$. Also, let $\mathbf{S}_2$ consist of the relation symbols $St(Sname, Sage, Saddress)$, $Cr(Cid, Cname, Pname)$ and $Take(Sname, Cid, Cgrade)$.

Relation $Student$ (St) stores students name and age (and address) information. Relation $Course$ (Cr) stores courses ids and names information, in addition to the program name which provides each course. Finally, relation $Enroll$ $(Take)$ stores the set of courses that each student completed.

Further, assume that $\mathbf{S}_1$ and $\mathbf{S}_2$ are connected by a mapping table $\mathcal{M}$ that consists of the following pairs $\{(CSI1390, ECOR1606), (CSI1390, COMP1005), (CS, CS), (ENG, ENG)\}$.

Let I be an instance of $\mathbf{S}_1$ and $J = \{St\ (Alex, 18, Ottawa), Cr\ (ECOR1606,$ *Problem Solving and Computers*, $ENG), Cr\ (COMP1005,$ *Introduction to Computer Science I*, $CS), Take\ (Alex, ECOR1606, 80)\}$ be an instance of $\mathbf{S}_2$.

According to [2], posing a query q to I that computes the list of students considered to have finished CS courses in UOO, will re-write q to a query q' to retrieve a similar list from UOC following the semantics of $\mathcal{M}$. A query q' can

be the following: q': Select $Sname$ From Cr, $Take$ Where $Cr.Cid = Take.Cid$ And $Cr.Pname = 'CS'$. In this case, the answer of posing q' to J is $\emptyset$. □

Assume that UOO accredits a 'CS' course to a student doing program transfer from UOC only if this student finishes an equivalent 'CS' course, according to $\mathcal{M}$, in UOC. In Example 1, *Alex* is not considered as finished a 'CS' course at UOC. Therefore, if *Alex* does a transfer to the CS program in UOO, he will not be credited the $CSI1390$ course. However, if the semantics of the mapping table $\mathcal{M}$ in this example specify that course $CSI1390$ in UOO is equivalent to the ENG course $ECOR1606$ in UOC, and course $CSI1390$ in UOO is equivalent to the CS course $COMP1005$ in UOC, then it can be deduced that courses $ECOR1606$ and $COMP1005$ are considered equivalent with respect to UOO according to $\mathcal{M}$. Therefore, given the fact that $Take$ (*Alex*, $ECOR1606$, 80) $\in J$ in Example 1 along with the equivalence semantics in $\mathcal{M}$, *Alex* is considered to have finished the equivalent CS course $COMP1005$ in UOC and he should be credited the 'CS' course $CSI1390$ with a grade 80 if he did a transfer to UOO.

To solve such a problem, we introduce a new class of settings, called *data sharing and exchange* (DSE) settings, where exchange occurs between a source and a target that use different sets of vocabularies. Despite the importance of the topic, the fundamentals of this process have not been laid out to date. In this paper, we embark on the theoretical foundations of such problem, that is, exchanging data between two independent applications with different sets of domains of constants. DSE settings extend DE settings with a *mapping table* $\mathcal{M}$, introduced in [9], to allow collaboration at the instance level. In addition, the set of source-to-target dependencies Σ_{st} in DSE refers to such mapping table so that coordination of distinct vocabularies between applications takes place together with the exchange.

From what we have mentioned so far about DSE, one would think that all DSE instances can be reduced to a usual DE instance where the source schema is extended with a mapping table $\mathcal{M}$. However, we argue in this paper that there exist DSE settings with particular interpretation of related data in mapping tables where DSE is different than a DE setting (as we show later in Example 2). One such particular interpretation of related data that we consider in this paper is: a source element is mapped to a target element only if both are considered to be equivalent (i.e. denote the same object). In this DSE scenario, DSE and DE are different because source and target data can be incomplete with respect to the "implicit" information provided by the semantics of mapping tables. To formalize this idea we use techniques developed by Arenas et al. in [5], where authors introduced a *knowledge exchange* framework for exchanging knowledge bases. It turns out that this framework suits our requirements, and in particular, allows us to define the exchange of both explicit and implicit data from source to target. Our main contributions in this work are the following:

(1) Universal DSE Solutions. We formally define the semantics of a DSE setting and introduce the class of universal DSE solutions, that can be seen as a natural generalization of the class of universal data exchange solutions [7] to the DSE scenario, and thus, as "good" solutions. A universal DSE solution consists

of a subset of explicit data that is necessary to infer the remaining implicit information using a given set Σ_t of rules in the target.

(2) Minimal Universal DSE Solutions. We define the class of minimal universal DSE solutions which are considered as "best" solutions. A minimal universal DSE solution contains the minimal amount of explicit data required to compute the complete set of explicit and implicit data using a set of target rules Σ_t. We show that there exists an algorithm to generate a *canonical minimal universal* DSE solution, with a well-behaved set Σ_t of target rules, in LOGSPACE.

(3) Query Answering. We formally define the set of DSE certain answers for conjunctive queries. We also show how to compute those efficiently using canonical minimal universal DSE solutions.

2 Preliminaries

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 symbol 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. Further, every time that we have two disjoint schemas $\mathbf{R}$ and $\mathbf{S}$, an instance I of $\mathbf{R}$ and an instance J of $\mathbf{S}$, we define (I, J) as the instance K of schema $\mathbf{R} \cup \mathbf{S}$ such that $R^K = R^I$, for each $R \in \mathbf{R}$, and $S^K = S^J$, for each $S \in \mathbf{S}$.

Data Exchange Settings. As is customary in the data exchange literature [7,8], we consider instances with two types of values: constants and nulls.[1] More precisely, let Const and Var be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathsf{D} = \mathsf{Const} \cup \mathsf{Var}$. If we refer to a schema $\mathbf{S}$ as a *source* schema, then we assume that for an instance I of $\mathbf{S}$, it holds that $dom(I) \subseteq \mathsf{Const}$; that is, source instances are assumed to be "complete", as they do not contain missing data in the form of nulls. On the other hand, if we refer to a schema $\mathbf{T}$ as a *target* schema, then for every instance J of $\mathbf{T}$, it holds that $dom(J) \subseteq \mathsf{Const} \cup \mathsf{Var}$; that is, target instances are allowed to contain null values.

A *data exchange* (DE) *setting* is a tuple $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, $\mathbf{S}$ and $\mathbf{T}$ do not have predicate symbols in common, and Σ_{st} consists of a set of *source-to-target tuple-generating* dependencies (st-tgds) that establish the relationship between source and target schemas. An st-tgd is a FO-sentence of the form:

$$\forall \bar{x} \forall \bar{y} \, (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z})), \tag{1}$$

[1] We usually denote constants by lowercase letters $a, b, c, \ldots$, and nulls by symbols $\perp, \perp', \perp_1, \ldots$

where $\phi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{S}$ and $\psi(\bar{x}, \bar{z})$ is a conjunction of relational atoms over $\mathbf{T}$.[2] A *source* (resp. *target*) instance K for $\mathfrak{S}$ is an instance of $\mathbf{S}$ (resp. $\mathbf{T}$). We usually denote source instances by $I, I', I_1, \ldots$, and target instances by $J, J', J_1, \ldots$.

An instance J of $\mathbf{T}$ is a *solution* for an instance I under $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, if the instance (I, J) of $\mathbf{S} \cup \mathbf{T}$ satisfies every st-tgd in Σ_{st}. If $\mathfrak{S}$ is clear from the context, we say that J is a solution for I.

The data exchange literature has identified a class of preferred solutions, called the *universal* solutions, that in a precise way represents all other solutions. In order to define these solutions, we need to introduce 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 $h : dom(K_1) \to dom(K_2)$ 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 for I under $\mathfrak{S}$. Then J is a *universal* solution for I under $\mathfrak{S}$, if for every solution J' for I under $\mathfrak{S}$, there exists a homomorphism from J to J'.

For the class of data exchange settings that we referred to in this paper, every source instance has a universal solution [7]. Further, given a DE setting $\mathfrak{S}$, there is a procedure (based on the *chase* [6]) that computes a universal solution for each source instance I under $\mathfrak{S}$. In the case when $\mathfrak{S}$ is fixed such procedure works in LOGSPACE. Assuming $\mathfrak{S}$ to be fixed is a usual and reasonable assumption in data exchange [7], as mappings are often much smaller than instances. We stick to this assumption for the rest of the paper.

Mapping Tables. Coordination can be incorporated at the *data* level, through the use of *mapping* tables [9]. These mechanisms were introduced in data coordination settings [2] to establish the correspondence of related information in different domains. In its simplest form, mapping tables are just binary tables containing pairs of corresponding identifiers from two different sources. Formally, given two domains D_1 and D_2, not necessarily disjoint, a mapping table over $(\mathsf{D}_1, \mathsf{D}_2)$ is nothing else than a subset of $\mathsf{D}_1 \times \mathsf{D}_2$. Intuitively, the fact that a pair (d_1, d_2) belongs to the mapping table implies that value $d_1 \in \mathsf{D}_1$ *corresponds* to value $d_2 \in \mathsf{D}_2$. Notice that the exact meaning of "correspondence" between values is unspecified and depends on the application.

In this paper we deal with a very particular interpretation of the notion of correspondence in mapping tables. We assume that the fact that a pair (a, b) is in a mapping table implies that a and b are equivalent objects. We are aware of the fact that generally mapping tables do not interpret related data in this way. However, we argue that this particular case is, at the same time, practically relevant (e.g. in peer-to-peer settings [9]) and theoretically interesting (as we will see along the paper).

This particular interpretation of mapping tables implies that they may contain implicit information that is not explicitly listed in their extension. For instance,

[2] We usually omit universal quantification in front of st-tgds and express them simply as $\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$.

assume that $\mathcal{M}$ is a mapping table that consists of the pairs $\{(a, c), (b, c), (b, d)\}$. Since a and c are equivalent, and the same is true about b and c, we can *infer* that a and b are equivalent. Also, we can infer using the same reasoning that c and d are equivalent. Such implicit information is, of course, valuable, and cannot be discarded at the moment of using the mapping table as a coordination tool. In particular, we will use this view of mapping tables as being incomplete with respect to its implicit data when defining the semantics of DSE settings.

3 Data Sharing and Exchange Settings

We formally define in this section DSE settings that extend DE settings to allow collaboration via mapping tables.

Definition 1 (DSE setting). *A* data sharing and exchange (*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 mapping (*we call the first attribute of* $\mathcal{M}$ *the source attribute and the second one the target attribute*)*; and* (3) Σ_{st} *consists of a set of* mapping *st-tgds, which are* FO *sentences of the form*

$$\forall \bar{x} \forall \bar{y} \forall \bar{z}\, (\phi(\bar{x}, \bar{y}) \wedge \mu(\bar{x}, \bar{z}) \rightarrow \exists \bar{w}\, \psi(\bar{z}, \bar{w})), \tag{2}$$

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* $\mathcal{M}$*,* (*iii*) $\bar{x}$ *is the tuple of variables that appear in* $\mu(\bar{x}, \bar{z})$ *in the positions of source attributes of* $\mathcal{M}$*, and* (*iv*) $\bar{z}$ *is the tuple of variables that appear in* $\mu(\bar{x}, \bar{z})$ *in the positions of target attributes of* $\mathcal{M}$*.*

We provide some terminology and notations before explaining the intuition behind the different components of a DSE setting. As before, instances of $\mathbf{S}$ (resp. $\mathbf{T}$) are called source (resp. target) instances, and we denote source instances by $I, I', I_1, \ldots$ and target instances by $J, J', J_1, \ldots$. Instances of $\mathcal{M}$ are called *source-to-target* mapping tables (st-mapping tables). By slightly abusing notation, we denote st-mapping tables also by $\mathcal{M}$.

Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting. We distinguish between the set of source constants, denoted by $\mathsf{Const}^{\mathbf{S}}$, and the set of target constants, denoted by $\mathsf{Const}^{\mathbf{T}}$, since applications that collaborate on data usually have different data domains. As in the case of usual data exchange, we also assume the existence of a countably infinite set Var of labelled nulls (that is disjoint from both $\mathsf{Const}^{\mathbf{S}}$ and $\mathsf{Const}^{\mathbf{T}}$). Also, in a DSE the domain of a source instance I is contained in $\mathsf{Const}^{\mathbf{S}}$, while the domain of a target instance J belongs to $\mathsf{Const}^{\mathbf{T}} \cup \mathsf{Var}$. On the other hand, the domain of the st-mapping table $\mathcal{M}$ is a subset of $\mathsf{Const}^{\mathbf{S}} \times \mathsf{Const}^{\mathbf{T}}$. Thus, coordination between the source and the target at the data level occurs when $\mathcal{M}$ identifies which source and target constants denote the same object. The intuition behind usual st-tgds is that they specify how source data has to be transformed to conform to the target schema (that is, coordination at the

schema level). However, since in the DSE scenario we are interested in transferring data based on the source instance as well as on the correspondence between source and target constants given by the st-mapping table that interprets $\mathcal{M}$, the mapping st-tgds extend usual st-tgds with a conjunction μ that filters the target data that is related via $\mathcal{M}$ with the corresponding source data.

More formally, given a source instance I and an st-mapping table $\mathcal{M}$, the mapping st-tgd $\phi(\bar{x}, \bar{y}) \wedge \mu(\bar{x}, \bar{z}) \rightarrow \exists \bar{w} \psi(\bar{z}, \bar{w})$ enforces the following: whenever $I \models \phi(\bar{a}, \bar{b})$, for a tuple $(\bar{a}, \bar{b})$ of constants in $\mathsf{Const}^{\mathbf{S}} \cap dom(I)$, and the tuple $\bar{c}$ of constants in $\mathsf{Const}^{\mathbf{T}}$ is related to $\bar{a}$ via μ (that is, $\mathcal{M} \models \mu(\bar{a}, \bar{c})$), then it must be the case that $J \models \psi(\bar{c}, \bar{d})$, for some tuple $\bar{d}$ of elements in $dom(J) \cap (\mathsf{Const}^{\mathbf{T}} \cup \mathsf{Var})$, where J is the materialized target instance. In usual DE terms, we should say that J is a solution for I and $\mathcal{M}$ under $\mathfrak{S}$, i.e. $(((I \cup \{\mathcal{M}\}), J) \vDash \Sigma_{st})$. However, as we see in the next section, solutions have to be defined differently in DSE. Therefore, to avoid confusions, we say J is a *pre-solution* for I and $\mathcal{M}$ under $\mathfrak{S}$.

Example 2. Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting. In reference to Example 1, assume that $\mathbf{S}$ in $\mathfrak{S}$ is the schema of UOC and $\mathbf{T}$ in $\mathfrak{S}$ is the schema of UOO.

Suppose that $\mathcal{M}$ in $\mathfrak{S}$ consists of the following pairs $\{(ECOR1606, CSI1390)$, $(COMP1005, CSI1390)$, $(COMP1005, CSI1790)$, (CS,CS), (ENG,ENG), $(Alex,Alex)$, $(18,18)\}$. Finally, let Σ_{st} consist of the following st-mapping dependencies:

$$(a)\ St(x,y,z) \wedge Take(x,w,u) \wedge Cr(w,v,\text{`}CS\text{'}) \wedge \mathcal{M}(x,x') \wedge \mathcal{M}(y,y') \rightarrow Student(x',y').$$

$$(b)\ St(x,y,z) \wedge Take(x,w,u) \wedge Cr(w,v,\text{`}CS\text{'}) \wedge \mathcal{M}(x,x') \wedge \mathcal{M}(w,w') \wedge \mathcal{M}(u,u') \rightarrow Enroll(x',w',u').$$

It is clear that this DSE instance is exchanging information of UOC students that have taken 'CS' courses with the list of courses they have finished. Also, $\mathcal{M}$ specifies that the *Introduction to Computers* course with $Cid = CSI1390$ in UOO has a French version course *Introduction aux Ordinateurs* with $Cid = CSI1790$ provided at UOO. Let $I = \{St(Alex, 18, Ottawa)$, $Cr(ECOR1606$, *Problem Solving and Computers*, $ENG)$, $Cr(COMP1005$, *Introduction to Computer Science I*, $CS)$, $Take$ $(Alex$, $ECOR1606$, $80)\}$ be an instance of $\mathbf{S}$. Then, $J = \emptyset$ is a pre-solution for I and $\mathcal{M}$ under $\mathfrak{S}$. □

We can see in Example 2 that in the pre-solution J, *Alex* is not considered as have finished a 'CS' course. However, if the st-mapping table $\mathcal{M}$ follows the semantics we adopt in this paper, then *Alex* should be considered to have completed the 'CS' course *Introduction to Computer Science I*. Therefore, we can easily deduce that in a DSE setting $\mathfrak{S}$, we cannot identify solutions with pre-solutions. One reason is that a source instance I in $\mathfrak{S}$ can be *incomplete* with respect to the semantics of $\mathcal{M}$ as the case in Example 2 above. A second reason is that data mappings in an st-mapping table $\mathcal{M}$ in $\mathfrak{S}$ can also be *incomplete* with respect to the semantics of $\mathcal{M}$ as we shall show in Sect. 4. Data mappings in an st-mapping table $\mathcal{M}$ are usually specified by domain specialists. However, $\mathcal{M}$

should record not only the associations suggested by the domain specialists, but also the ones inferred by its semantics. Therefore, to capture the real semantics of the DSE problem, we came up with a more sophisticated notion of a solution that we introduce in the following section.

4 DSE and Knowledge Exchange

From now on we use the equivalence relation $\sim$ as $a \sim b$ to intuitively denote that a and b, where $\{a, b\} \subseteq \mathsf{Const}^{\mathbf{S}}$ (or $\{a, b\} \subseteq \mathsf{Const}^{\mathbf{T}}$) are *inferred* by the semantics of an st-mapping table $\mathcal{M}$ as equivalent objects. Let us revisit Example 2. There are two ways in which the data in $\mathfrak{S}$ is incomplete: First of all, since $\mathcal{M}(ECOR1606, CSI1390)$ holds in $\mathfrak{S}$, then UOC course $ECOR1606$ is equivalent to the UOO course $CSI1390$. Also, since $\mathcal{M}(COMP1005, CSI1390)$ holds, then UOC course $COMP1005$ is equivalent to the UOO course $CSI1390$. Therefore, we can deduce that $ECOR1606 \sim COMP1005$ with respect to the target UOO. This means, according to semantics of $\sim$, the source instance I is incomplete, since I should include the tuple $Take(Alex, COMP1005, 80)$ in order to be complete with respect to $\mathcal{M}$.

Second, since $\mathcal{M}(COMP1005, CSI1390)$ holds in $\mathfrak{S}$, then the UOC course $COMP1005$ is equivalent to the UOO course $CSI1390$ according to the semantics of $\mathcal{M}$. Also, since $\mathcal{M}(COMP1005, CSI1790)$ holds in $\mathfrak{S}$, then course $COMP1005$ is equivalent to the UOO course $CSI1790$. Therefore, we can deduce that $CSI1390 \sim CSI1790$, according to the semantics of $\mathcal{M}$. This implies that $\mathcal{M}$ is incomplete, since the fact that $\{(ECOR1606, CSI1390), (COMP1005, CSI1390), (COMP1005, CSI1790)\} \subseteq \mathcal{M}$ entails from the semantics of $\sim$ the fact that $(ECOR1606, CSI1790) \in \mathcal{M}$. Therefore, we say I and $\mathcal{M}$ are incomplete in the sense that they do not contain all the data that is implied by the semantics of $\mathcal{M}$. Further, it is not hard to see that the completion process we just sketched can become recursive in more complex DSE instances.

Given the above reasoning, again one would think to solve the DSE problem by reducing it to a DE setting $\mathfrak{S}$ with a set of dependencies defined over a combined schema $(\mathbf{S} \cup \{\mathcal{M}\})$, which complete the source instance I and the st-mapping table $\mathcal{M}$ under $\mathfrak{S}$ with the additional data entailed by the semantics of $\mathcal{M}$. However, usually the real reason behind defining dependencies over a databases schema is to ensure that data stored in the extension of this schema or new coming tuples follow the structure of it [16]. Also, such constraints are not treated as additional *implicit* data that represents specific semantics in st-mapping tables, and whose purpose is to entail new facts in addition to the stored ones. Therefore, to apply the intuition explained in Example 2 and to generate "good" solutions in a DSE setting, it is assumed to be fundamental that both explicit data stored in I and implicit data entailed by the semantics of $\mathcal{M}$ are exchanged to the target. Contrarily, we show in what follow that semantics of a DSE setting vary from those of DE setting since there exist solutions which consist of a portion of the fully exchanged set of data and yet are still good solutions that proved in Sect. 7 to be efficient for conjunctive query answering.

From what we explained so far, we conclude that the real semantics of a DSE setting is based on the explicit data contained in I and $\mathcal{M}$, in addition to the implicit data obtained by following a completion process for the source, the target, and $\mathcal{M}$.

We define below a set of FO sentences, of type full tgds[3], over a schema $\mathbf{S} \cup \mathcal{M}$ ($\mathbf{T} \cup \mathcal{M}$) extended with a fresh binary relation symbol EQUAL that appears neither in $\mathbf{S}$ nor in $\mathbf{T}$ and that captures the semantics of $\sim$ in a recursive scenario, which formally defines this completion process:

Definition 2 (Source and Target Completion). *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting. The* source completion *of $\mathfrak{S}$, denoted by Σ_s^c, is the conjunction of the following* FO *sentences over the schema $\mathbf{S} \cup \{\mathcal{M}, \text{EQUAL}\}$:*

1. *For each $S \in \mathbf{S} \cup \{\mathcal{M}\}$ of arity n and $1 \leq i \leq n$: $\forall x_1 \cdots \forall x_n (S(x_1, \ldots, x_i, \ldots, x_n) \rightarrow \text{EQUAL}(x_i, x_i))$.*
2. $\forall x \forall y (\text{EQUAL}(y, x) \rightarrow \text{EQUAL}(x, y))$.
3. $\forall x \forall y \forall z (\text{EQUAL}(x, z) \wedge \text{EQUAL}(z, y) \rightarrow \text{EQUAL}(x, y))$.
4. $\forall x \forall y \forall z (\mathcal{M}(x, z) \wedge \mathcal{M}(y, z) \rightarrow \text{EQUAL}(x, y))$.
5. $\forall x \forall y \forall z \forall w (\mathcal{M}(x, z) \wedge \text{EQUAL}(x, y) \wedge \text{EQUAL}(z, w) \rightarrow \mathcal{M}(y, w))$.
6. *For each $S \in \mathbf{S}$ of arity n: $\forall x_1, y_1 \cdots \forall x_n, y_n (S(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i) \rightarrow S(y_1, \ldots, y_n))$.*

The target completion *of $\mathfrak{S}$, denoted Σ_t^c, is defined analogously by simply replacing the role of $\mathbf{S}$ by $\mathbf{T}$ in Σ_s^c, and then adding the rule* 7. $\forall x \forall y \forall z (\mathcal{M}(z, x) \wedge \mathcal{M}(z, y) \rightarrow \text{EQUAL}(x, y))$ *that defines the completion of $\mathcal{M}$ over the target.*

Notice that the first 3 rules of Σ_s^c make sure that EQUAL is an equivalence relation on the domain of the source instance. The fourth rule detects which source elements have to be declared equal by the implicit knowledge contained in the st-mapping table. The last two rules allow to complete the interpretation of $\mathcal{M}$ and the symbols of $\mathbf{S}$, by adding elements declared to be equal in EQUAL. The intuition for Σ_t^c is analogous.

Summing up, data in a DSE scenario always consists of two modules: (1) The explicit data stored in the source instance I and the st-mapping table $\mathcal{M}$, and (2) the implicit data formalized in Σ_s^c and Σ_t^c. This naturally calls for a definition in terms of *knowledge exchange* [5], as defined next. A knowledge base (KB) over 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). The knowledge base representation has been used to represent various types of data including ontologies in the semantic web, which are expressed using different types of formalisms including *Description Logic* (DL) [12].

The set of *models* of (K, Σ) [5], denoted by $\mathsf{Mod}(K, \Sigma)$, is defined as the set of instances of $\mathbf{R}$ that contain the explicit data in K and satisfy the implicit data in Σ; that is, $\mathsf{Mod}(K, \Sigma)$ corresponds to the set $\{K' \mid K' \text{ is an instance of } \mathbf{R}, K \subseteq K' \text{ and } K' \models \Sigma\}$. In DSE, we consider source KBs of the form $((I \cup \{\mathcal{M}\}), \Sigma_s^c)$, which intuitively correspond to completions of the source instance

[3] Full tgds are tgds that do not use existential quantication.

I with respect to the implicit data in $\mathcal{M}$, and, analogously, target KBs of the form $((J \cup \{\mathcal{M}\}), \Sigma_t^c)$.

A good bulk of work has recently tackled the problem of exchange of KBs that are defined using different DL languages [4]. we formalize the notion of (universal) DSE solution to extend the KB (universal) solution introduced in [5]. The main difference is that in DSE solutions we need to coordinate the source and target information provided by $\mathcal{M}$, as opposed to KB solutions that require no data coordination at all. This is done by establishing precise relationships in a (universal) DSE solution between the interpretation of $\mathcal{M}$ in $\mathbf{S}$ and $\mathbf{T}$, respectively. KB exchange in DL showed that target KB (universal) solutions [5] present several limitations since these can miss some semantics of the source KB [4]. Universal DSE solutions, on the other hand, do not have those limitations and they reflect the semantics in the source and the st-mapping table accurately.

From now on, $K_{\mathbf{R}'}$ denotes the restriction of instance K to a subset $\mathbf{R}'$ of its schema $\mathbf{R}$. Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a target instance. Recall that Σ_s^c, Σ_t^c are the source and target completions of $\mathfrak{S}$, respectively. Then:

1. J is a *DSE solution* for I and $\mathcal{M}$ under $\mathfrak{S}$, if for every $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^c)$ there is $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}), \Sigma_s^c)$ such that the following hold: (a) $K'_{\mathcal{M}} \subseteq K_{\mathcal{M}}$, and (b) $K_{\mathbf{T}}$ is a pre-solution for $K'_{\mathbf{S}}$ and $K'_{\mathcal{M}}$ under $\mathfrak{S}$.
2. 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 $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}, \Sigma_s^c)$ there is $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}, \Sigma_t^c)$ such that (a) $K_{\mathcal{M}} \subseteq K'_{\mathcal{M}}$, and (b) $K_{\mathbf{T}}$ is a pre-solution for $K'_{\mathbf{S}}$ and $K'_{\mathcal{M}}$ under $\mathfrak{S}$.

In Example 2, $J = \{Student(Alex, 18)$, $Enroll$ $(Alex, CSI1390, 80)$, $Enroll$ $(Alex, CSI1790, 80)\}$ is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. We define below a simple procedure $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ that, given a DSE setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ and a source instance I and an st-mapping table $\mathcal{M}$, it generates a universal DSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.

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

1. Apply the source completion process, Σ_s^c, to I and $\mathcal{M}$, and generate $\hat{I}$ and $\hat{\mathcal{M}}$ respectively.
2. Apply a procedure (based on the *chase* [6]) to the instance $(\hat{I} \cup \{\hat{\mathcal{M}}\})$, and generate a canonical universal pre-solution J for $\hat{I}$ and $\hat{\mathcal{M}}$.

The procedure $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ works as follows: step 1 applies the source completion process Σ_s^c, given in Definition 2, to I and $\mathcal{M}$, and returns as outcome the source instance $\hat{I}$ and the st-mapping table $\hat{\mathcal{M}}$ that are complete with respect to the implicit data in $\mathcal{M}$. Next, step 2 generates a canonical universal pre-solution J for $\hat{I}$ and $\hat{\mathcal{M}}$ such that $((\hat{I} \cup \hat{\mathcal{M}}), J) \vDash \Sigma_{st}$.

We can combine the fact that universal solutions in fixed data exchange settings $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ can be computed in LOGSPACE [3] with some deep

results in the computation of symmetrical binary relations [10], to show that universal DSE solutions can be computed in LOGSPACE:

Proposition 1. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE setting. Then computing a universal DSE solution J for a source instance I and an st-mapping table $\mathcal{M}$ is in* LOGSPACE.

Proof. To prove that the $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ procedure computes a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$ in LOGSPACE, we rely on the following facts:

(1) In Σ_s^c (in step 1 to step 4), we compute the transitive closures of the *symmetrical* binary table EQUAL. Computing the transitive closure of symmetrical binary relations is solvable in LOGSPACE [10]; (2) steps 5 and 6 of Σ_s^c compute the complete instances $\hat{I}$ and the st-mapping table $\hat{\mathcal{M}}$ using EQUAL. This step can be computed by applying the naive chase procedure to I and $\mathcal{M}$ using rules 5 and 6 of Σ_s^c. Following the result in [7] that a naive chase procedure runs in LOGSPACE, we can deduce that generating $\hat{I}$ and $\hat{\mathcal{M}}$ using steps 5 and 6 of Σ_s^c is in LOGSPACE; (3) step 2 of $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ procedure can be reduced to the problem of computing a universal solution for $(\hat{I} \cup \{\hat{\mathcal{M}}\})$ in a fixed DE setting $\mathfrak{S}$. Consequently, similar to steps 5 and 6 in Σ_s^c, this process works in LOGSPACE in a fixed DSE setting; (4) finally, since LOGSPACEis closed under composition [11], we conclude that $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ is in LOGSPACE.

5 Minimal Universal DSE Solutions

In the context of ordinary data exchange, "best"solutions – called cores – are universal solutions with minimal size. In knowledge exchange, on the other hand, "best" solutions are cores that materialize a minimal amount of explicit data. Intuitively, a *minimal* universal DSE (MUDSE) solution is a core universal DSE solution J that contains a minimal amount of explicit data in J with respect to Σ_t^c, and such that no universal DSE solution with strictly less constants is also a universal DSE solution with respect to Σ_t^c.

We define this formally: Let $\mathfrak{S}$ be a DSE setting, I be a source instance, $\mathcal{M}$ an st-mapping table, and J a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Then J is a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, if: (1) There is no proper subset J' of J such that J' is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, and; (2) There is no universal DSE solution J' such that $dom(J') \cap \mathsf{Const}^{\mathbf{T}}$ is properly contained in $dom(J) \cap \mathsf{Const}^{T}$.

So, in Example 2, $J = \{Student(Alex, 18)$, $Enroll\ (Alex, CSI1390, 80)\}$ is a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Note that the DSE solution $J' = \{Student(Alex, 18)$, $Enroll\ (Alex, CSI1390, 80), Enroll\ (Alex, CSI1790, 80)\}$ is a core universal DSE solution, however it is not the most compact one. Condition (2) in the definition of MUDSE solutions is not part of the original definition of minimal solutions in knowledge exchange [5]. However, this condition is necessary as we see below.

Assume that the universal DSE solution in Example 2 includes the following two facts $\{Teach(Anna, CSI1390), Teach(Anna, CSI1790)\}$, where $\mathbf{T}$

is extended with the relation $Teach(Tid, Cid)$ which specifies the teachers and the list of courses they teach. Then, the DSE solution $J = \{Student(Alex, 18), Enroll\ (Alex, CSI1390, 80), Teach(Anna, CSI1790)\}$ does not satisfy condition (2) and provides us with redundant information with respect to I and $\mathcal{M}$, since we can conclude that $CSI1390$ and $CSI1790$ are equivalent courses. A MUDSE solution however would be $J = \{Student(Alex, 18), Enroll(Alex, CSI1390, 80), Teach(Anna, CSI1390)\}$.

We define below a procedure $\texttt{CompMUDSEsol}_{\mathfrak{S}}$, that given a DSE setting $\mathfrak{S}$, a source instance I, and an st-mapping table $\mathcal{M}$, it computes a MUDSE solution J^* for I and $\mathcal{M}$ under $\mathfrak{S}$ in LOGSPACE. This procedure works as follows:
$\texttt{CompMUDSEsol}_{\mathfrak{S}}$:
Input: A source instance I, an st-mapping table $\mathcal{M}$, and a set Σ_{st} of st-tgds.
Output: A Minimal Universal DSE solution J^* for I and $\mathcal{M}$ under $\mathfrak{S}$.

1. Apply the source completion process, Σ_s^c, to I and $\mathcal{M}$, and generate $\hat{I}$ and $\hat{\mathcal{M}}$ respectively.
2. Define an equivalence relation $\sim$ on $dom(\hat{\mathcal{M}}) \cap \mathsf{Const}^{\mathbf{T}}$ as follows: $c_1 \sim c_2$ iff there exists a source constant a such that $\hat{\mathcal{M}}(a, c_1)$ and $\hat{\mathcal{M}}(a, c_2)$ hold.
3. Compute equivalence classes $\{C_1, \ldots, C_m\}$ for $\sim$ over $dom(\hat{\mathcal{M}}) \cap \mathsf{Const}^{\mathbf{T}}$ such that c_1 and c_2 exist in C_i only if $c_1 \sim c_2$.
4. Choose a set of witnesses $\{w_1, \ldots, w_m\}$ where $w_i \in C_i$, for each $1 \leq i \leq m$.
5. Compute from $\hat{\mathcal{M}}$ the instance $\mathcal{M}_1 := \texttt{replace}(\hat{\mathcal{M}}, w_1, \ldots, w_m)$ by replacing each target constant $c \in C_i \cap dom(\hat{\mathcal{M}})$ $(1 \leq i \leq m)$ with $w_i \in C_i$.
6. Apply a procedure (based on the *chase* [6]) to the instance $(\hat{I} \cup \{\mathcal{M}_1\})$, and generate a canonical universal pre-solution J for $\hat{I}$ and $\mathcal{M}_1$.
7. Apply a procedure (based on the core [8]) to the target instance J and generate the target instance J^* that is the core of J.

We prove the correctness of $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ in the following Theorem.

Theorem 1. *Let $\mathfrak{S}$ be a DSE setting, I a source instance, and $\mathcal{M}$ an st-mapping table. Suppose that J^* is an arbitrary result for $\texttt{CompMUDSEsol}_{\mathfrak{S}}(I, \mathcal{M})$. Then, J^* is a minimal universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.*

In data exchange, the smallest universal solutions are known as *cores* and can be computed in LOGSPACE [8]. With the help of such result we can prove that MUDSE solutions can be computed in LOGSPACE too. Also, in this context MUDSE solutions are unique up to isomorphism:

Theorem 2. *Let $\mathfrak{S}$ be a fixed DSE setting. There is a* LOGSPACE *procedure that computes, for a source instance I and an st-mapping table $\mathcal{M}$, a MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$. Also, for any two MUDSE solutions J_1 and J_2 for I and $\mathcal{M}$ under $\mathfrak{S}$, it is the case that J_1 and J_2 are isomorphic.*

Proof. The proof of the first part of this theorem is very similar to the proof of Proposition 1, with the difference that steps 2, 3, and 4 in $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ seem to be non-deterministic since they involve choosing a set of witnesses $\{w_1, \ldots, w_m\}$ for $\{C_1, \ldots, C_m\}$. Clearly, different sets of witnesses may yield different

target instances. However, each possible choice of witnesses leads to a minimal universal DSE solution. In addition, according to [7,8], generating cores can be computed in LOGSPACE by applying the *naive* chase and the simple *Greedy* algorithm [8]. Finally, since LOGSPACE is closed under composition [11], we can deduce that the procedure $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ is computed in LOGSPACE.

6 Query Answering

In data exchange, one is typically interested in the *certain answers* of a query Q, that is, the answers of Q that hold in each possible solution [7]. For the case of DSE we need to adapt this definition to solutions that are knowledge bases. Formally, let $\mathfrak{S}$ be a DSE setting, I a source instance, $\mathcal{M}$ an st-mapping table, and Q a FO conjunctive query over $\mathbf{T}$. The set of certain answers of Q over I and $\mathcal{M}$ and under $\mathfrak{S}$, denoted $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, corresponds to the set of tuples that belong to the evaluation of Q over $K_{\mathbf{T}}$, for each DSE solution J for I and $\mathcal{M}$ and $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^c)$.

Example 3. We refer to the DSE setting given in Example 2. Let $Q(x, y, z) = Enroll(x, y, z)$. Then, $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \{Enroll(Alex, CSI1390, 80), Enroll(Alex, CSI1790, 80)\}$. □

In DE, certain answers of unions of CQs can be evaluated in LOGSPACE by directly posing them over a universal solution [7], and then discarding tuples with null values. The same complexity bound holds in DSE by applying a slightly different algorithm. In fact, certain answers cannot be simply obtained by posing Q on a universal DSE solution J, since J might be incomplete with respect to the implicit data in Σ_t^c.

One possible solution would be to apply the target completion program Σ_t^c to a universal DSE solution J (denoted as $\Sigma_t^c(J)$) as a first step, then apply Q to $\Sigma_t^c(J)$. A second method is to compute certain answers of Q using a MUDSE solution. A MUDSE solution J in DSE possesses an interesting property, that is, applying Q to J returns a set of certain answers U that minimally represents the set of certain answers U' returned when Q is applied to $\Sigma_t^c(J)$. We can compute $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ directly using J, by first applying rules in Σ_t^c, excluding rule 6, to generate the binary table EQUAL. Then complete the evaluation of Q on J, $Q(x_1, \ldots, x_n)$, and return $\hat{Q}(y_1, \ldots, y_n) = Q(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i)$.

Adopting the second method to compute certain answers using MUDSE solutions and EQUAL proved in Sect. 7 to exhibit a much better performance in run times than the first method. These results make MUDSE solutions distinguished for their compactness and for their performance in query answering. We also obtain the following result:

Proposition 2. *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a MUDSE solution, and Q a fixed CQ over $\mathbf{T}$. Then, $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \hat{Q}(J)$ where $\hat{Q}(y_1, \ldots, y_n) = Q(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i)$*

In addition, we prove in the following proposition that we can still compute the set of certain answers of a conjunctive query Q in LOGSPACE.

Proposition 3. *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ *be a fixed DSE setting and* Q *a fixed union of CQs. There is a* LOGSPACE *procedure that computes* $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$*, given a source instance* I *and an st-mapping table* $\mathcal{M}$*.*

Proof. Let Q be a fixed union of conjunctive queries. The fact that computing the set of certain answers of Q, $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ is in LOGSPACE, is based on the following facts: (1) following Proposition 1, generating a universal DSE solution is in LOGSPACE; and (2) following Theorem 2, generating a MUDSE solution is in LOGSPACE; and finally (3) it is known from [11] that the data complexity of any FO formula is in LOGSPACE, and thus checking if a fixed conjunctive query is satisfied in a database instance is in LOGSPACE.

7 Experiments

We implement the knowledge exchange semantics we introduced in this paper in a DSE prototype system. This system effectively generates universal DSE and MUDSE solutions that can be used to compute certain answers for CQs using the two methods introduced in Sect. 6. We used the DSE scenario of Example 2 extended with the st-tgd: $Cr(x, y, z) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y') \wedge \mathcal{M}(z, z') \rightarrow Course(x', y', z')$. Due to the lack of a benchmark that enforces recursion of the $\sim$ equivalence relation in the st-mapping table $\mathcal{M}$ and due to size restrictions, we synthesized the data in our experiments.

We show in our experiments that as the percentage of recursion increases in an st-mapping table, the run time to generate a universal DSE solution exceeds the time to generate a MUDSE solution. We also show that computing certain answers using a MUDSE solution is more effective than using a universal DSE solution. The experiments were conducted on a Lenovo workstation with a Dual-Core Intel(R) 1.80 GHz processor running Windows 7, and equipped with 4GB of memory and a 297 GB hard disk. We used Python (v2.7) to write the code and PostgreSQL (v9.2) database system.

DSE and MUDSE Solutions Computing Times. We used in this experiment a source instance I of 4,500 tuples, and 500 of those were courses information. The DSE system leveraged the work done in the state of the art ++Spicy system [13] to generate MUDSE solutions. We mapped courses data in the source to common target courses in $\mathcal{M}$, with different $\sim$ equivalence percentages (to enforce a recursive $\sim$ relation). The remaining set of source data was mapped to itself in $\mathcal{M}$. Figure 1 shows that as the percentage of recursion in $\sim$ equivalence relation over $\mathcal{M}$ increases, the run times to generate universal DSE and MUDSE solutions increase. The reason is, as the $\sim$ percentage increases, the number of source values (and target values) inferred to be $\sim$ increases, and thus the size of EQUAL created in Σ_s^c and Σ_t^c increases. Also, since target instances are usually larger than $\mathcal{M}$, the run time of completing the former to generate DSE solutions exceeds the time of completing the later when generating MUDSE solutions.

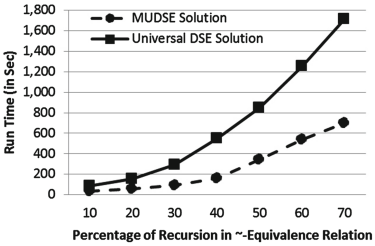

Fig. 1. MUDSE and Universal DSE solutions generation times

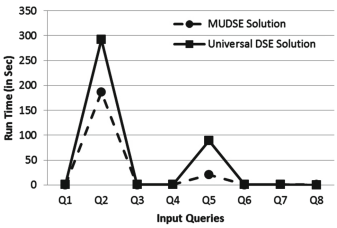

Fig. 2. Queries run times against a core of a universal DSE solution and a MUDSE solution

Table 1. List of queries

$Q1$	*Fetch all the students names and the name of courses they have taken*
$Q2$	*Fetch the list of pairs of students ids and names that took the same course*
$Q3$	*Fetch all the students names and the grades they have received*
$Q4$	*Fetch the list of pairs of courses names that belong to the same program*
$Q5$	*Fetch for each student id the pair of courses that he has finished with the same grade*
$Q6$	*Fetch all the courses ids and their names*
$Q7$	*Fetch all the students ids and their names*
$Q8$	*Fetch the list of pairs of students ids that possess the same address*

Conjunctive Queries Computing Times. We have selected a set of 8 queries to compare the performance of computing certain answers using a universal DSE solution (following the first method in Sect. 6) versus a MUDSE solution (following the second method in Sect. 6). We list the queries in Table 1.

We applied the list of input queries to a DSE instance where the $\sim$ percentage is 40 % and a course in the source is mapped to a maximum of two courses in the target. We chose a universal DSE solution, with a property of being a core of itself, that had around 18,000 records, and a MUDSE solution that contained around 4,900 records. Figure 2 shows that computing the sets of certain answers for the input conjunctive queries using a MUDSE solution take less run times than when computing these using a DSE solution. In addition, the deterioration in performance of query execution against the DSE solution appeared more in queries $Q2$ and $Q5$ than the remaining queries, is because both queries apply join operations to the *Enroll* table that involves a lot of elements which are inferred to be *equivalent* by $\mathcal{M}$.

8 Concluding Remarks

We introduced a DSE setting which exchanges data between two applications that have distinct schemas and distinct yet related sets of vocabularies. To cap-

ture the semantics of this setting, we defined DSE as a knowledge exchange system with a set of source and target rules that infer the implicit data should be in the target. We formally defined DSE solutions and identified the minimal among those. Also, we studied certain answers for CQs. Finally, we presented a prototype DSE system that generates universal DSE solutions and minimal ones, and it computes certain answers of CQs. In future work, we will investigate a more general DSE setting were mapped elements are not necessarily equal.

Acknowledgments. We thank NSERC for providing us the grants.

References

1. Bernstein, P., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serani, L., Zaihrayeu, I.: Data management for Peer-to-Peer computing: a vision. In: Proceedings of the Workshop on the Web and Databases (WebDB'02) (2002)
2. Arenas, M., Kantere, V., Kementsietsidis, A., Kiringa, I., Miller, R.J., Mylopoulos, J.: The hyperion project: from data integration to data coordination. In: ACM SIGMOD Record, pp. 53–58 (2003)
3. Arenas, M., Barceló, P., Libkin, L., Murlak, F.: Relational and XML Data Exchange. Morgan and Claypool Publishers, New York (2010)
4. Arenas, M., Botoeva, E., Calvanese, D.: Knowledge base exchange. In: Proceedings of Description Logics (2011)
5. Arenas, M., Perez, J., Reutter, J.L.: Data exchange beyond complete data. In: Proceedings of PODS, pp. 83–94 (2011)
6. Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. J. ACM **71**(4), 718–741 (1984)
7. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci **336**(1), 89–124 (2005)
8. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core. ACM Trans. Database Syst **30**(1), 174–210 (2005)
9. Kementsietsidis, A., Arenas, M., Miller, R.J.: Mapping data in peer-to-peer systems: semantics and algorithmic issues. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 325–336 (2003)
10. Reinghold, O.: Undirected connectivity in log-space. J. ACM **55**(4), 1–24 (2008)
11. Arenas, M., Reutter, J., Barceló, P.: Query languages for data exchange: beyond unions of conjunctive queries. In: Proceedings of the 12th International Conference on Database Theory, pp. 73–83 (2009)
12. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
13. Marnette, B., Mecca, G., Papotti, P.: ++Spicy: an open-source tool for second-generation schema mapping and data exchange. In: Proceedings of the VLDB, pp. 1438–1441 (2011)
14. Levy, A.Y., Rajaraman, A., Ordille, J.: Querying heterogeneous information sources using source descriptions. In: Proceedings of VLDB, pp. 251–262 (1996)
15. Larson, J.A., Sheth, A.P.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv **22**(3), 183–236 (1990)
16. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Semant. **7**(2), 74–89 (2009)

Springer

http://www.springer.com/978-3-319-08908-9

Declarative Programming and Knowledge Management
Declarative Programming Days, KDPD 2013, Unifying INAP, WFLP, and WLP, Kiel, Germany, September 11–13, 2013, Revised Selected Papers
Hanus, M.; Rocha, R. (Eds.)

2014, X, 251 p. 51 illus., Softcover
ISBN: 978-3-319-08908-9