Conceptual Schema Transformation in Ontology-based Data Access

Diego Calvanese[1], Tahir Emre Kalayci[2,1], Marco Montali[1], Ario Santoso[3,1], and Wil van der Aalst[4]

[1] KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano (Italy)
{calvanese,tkalayci,montali,santoso}@inf.unibz.it
[2] Virtual Vehicle Research Center, Graz (Austria)
emre.kalayci@v2c2.at
[3] Department of Computer Science, University of Innsbruck (Austria)
ario.santoso@uibk.ac.at
[4] Process and Data Science (PADS), RWTH Aachen University (Germany)
wvdaalst@pads.rwth-aachen.de

Abstract. Ontology-based Data Access (OBDA) is a by now well-established paradigm that relies on conceptually representing a domain of interest to provide access to relational data sources. The conceptual representation is given in terms of a *domain schema* (also called an ontology), which is linked to the data sources by means of declarative mapping specifications, and queries posed over the conceptual schema are automatically rewritten into queries over the sources. We consider the interesting setting where users would like to access the same data sources through a new conceptual schema, which we call the *upper schema*. This is particularly relevant when the upper schema is a reference model for the domain, or captures the data format used by data analysis tools. We propose a solution to this problem that is based on using transformation rules to map the upper schema to the domain schema, building upon the knowledge contained therein. We show how this enriched framework can be automatically transformed into a standard OBDA specification, which directly links the original relational data sources to the upper schema. This allows us to access data directly from the data sources while leveraging the domain schema and upper schema as a lens. We have realized the framework in a tool-chain that provides modeling of the conceptual schemas, a concrete annotation-based mechanism to specify transformation rules, and the automated generation of the final OBDA specification.

Keywords: Conceptual schema transformation, ontology-based data access, ontology-to-ontology mapping

1 Introduction

During the last two decades, (structural) conceptual schemas have been increasingly adopted not only to understand and document the relevant aspects of an application domain at a high level of abstraction, but also as live, computational artifacts. In particular, the paradigm of Ontology-Based Data Access (OBDA) exploits conceptual schemas (also called ontologies) as an intermediate layer for accessing and querying data stored

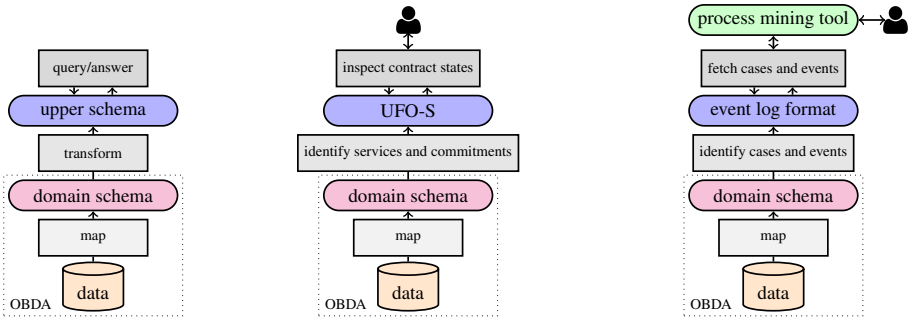
inside legacy information systems [23]. In the context of OBDA, the conceptual schema provides end-users with a vocabulary they are familiar with, at the same time masking how data are concretely stored, and enriching those (incomplete) data with domain knowledge. In this light, we call such a conceptual schema *domain schema*. The abstraction mismatch between the domain schema and the underlying data sources is covered by a second conceptual component, namely a mapping specification that declaratively links such two layers, expressing how patterns in the data correspond to domain concepts and relationships. Once an OBDA specification is in place, end-users may inspect the data sources by expressing high-level queries over the domain schema. An OBDA system handles this challenging setting by automatically rewriting such queries into corresponding low-level queries that are directly posed over the data sources, and by automatically reformulating the so-obtained answers into corresponding answers expressed over the domain schema. This supports domain experts in autonomously interacting with legacy data without the manual intervention of IT savvy. Notably, the actual data storage is completely transparent to end-users, who see the data in the form of conceptual objects and relations, *even though no actual materialization takes place*. From the foundational point of view, this is made possible by carefully tuning the expressive power of the conceptual modeling and mapping specification languages, and by exploiting key formal properties of their corresponding logic-based representations [4]. On top of these foundations, several OBDA systems have been engineered, *ontop* being one of the main representatives in this spectrum [3].[5]

OBDA has been subject of extensive research, and its advantages have been concretely shown in a plethora of application domains (see, e.g., [11,15,17]). Surprisingly, though, no research has been carried out on how to suitably extend the OBDA approach to handle the common situation where a higher-level conceptual schema (which we call *upper schema*) is needed to further abstract the knowledge captured by the domain schema. This happens when there is the need of viewing the domain schema and, in turn, the underlying data, according to a predefined structure, described by a reference model or an interchange format. In addition, different users may need to generate different views on the data, possibly using different upper schemas.

There are in particular two situations where the need for such a multi-level approach to data access is apparent. The first is when an OBDA specification is already in place, but certain types of users adopt reference models as an upper schema to understand the organization, create reports, and exchange information with external stakeholders. For example, the manager of an e-commerce company needs to reconstruct the state of contractual relationships and mutual commitments with customers, on top of the domain concepts of orders, payments, and deliveries. In this setting, which we discuss later on in the paper, the main objective is to reuse the available OBDA specification.

The second is the one where data analysis applications are exploited to extract insights from legacy data. Here, the main problem is that the actual input for such applications consists of specific abstractions that may not be explicitly present in the legacy data, and that have to be represented according to the expected input data format. E.g., the application of process mining techniques to discover, monitor, and improve business processes [1] requires structured event data as input using key notions such

[5] `http://ontop.inf.unibz.it`



(a) 2-level OBDA (b) 2OBDA framework for service management (c) 2OBDA framework for process mining

Fig. 1: The 2OBDA framework, and two relevant concrete instantiations

as case (process instance), event, timestamp, activity, etc. Only event data that meet these requirements can be mapped onto input formats like XES [14] and used by process mining tools. This is especially challenging in the common case where the organization relies on legacy information systems and/or general-purpose management systems (such as ERP or CRM applications) [1]. Previous approaches [8,6] tackled this problem by hard-coding the XES format inside the log extraction framework. While XES is supported by a wide range of tools (such as *Disco* by Fluxicon, *Celonis Process Mining* by Celonis, *minit* by Gradient ECM, and many more), it does not provide native support for important aspects such as durative and hierarchical events (such as those naturally provided by enterprise systems like SAP, and consequently supported in process mining tools like Celonis Process Mining). Accounting for these aspects would require flexibility in the event log format, that is, the possibility of tuning the upper event schema based on the application context and needs.

To tackle such challenging but common scenarios, we propose to suitably extend the OBDA framework so as to take into account multiple conceptual layers at once. We focus on the case where two conceptual layers are present, accounting for the domain and upper schemas, and call the resulting setting *2-level OBDA* (2OBDA for short). However, all our results seamlessly extend to the case where more layers are present.

Specifically, our contribution is threefold. First, in Section 4, we introduce the 2OBDA model as an elegant extension of OBDA. The core part of the framework is the *conceptual transformation* of concepts and relations in the domain schema into corresponding concepts and relations in the upper schema. This is specified in a declarative way, similarly to OBDA mappings but in this case accounting for ontology-to-ontology correspondences. The resulting framework (sketched in Figure 1a) is different from well-known approaches for ontology matching [12] and ontology/schema mapping [9], the first being focused on overlapping conceptual schemas that work at the same level of abstraction, the second neglecting the presence of underlying data.

Second, we show how a 2OBDA specification can be automatically compiled into a classical OBDA specification that directly connects the legacy data to the upper schema,

fully transparently to the end-users. Consequently, these can query the legacy data through the upper schema, by resorting to standard OBDA systems like *ontop*.

Finally, in Section 5, we report on how the approach has been realized in a tool-chain that supports end-users in modeling the domain and upper schema, and in specifying the corresponding transformations as annotations of the domain schema, whose types and features are derived from the concepts present in the upper schema. Notably, the tool-chain fully implements the compilation technique described above.

2 Motivation

In this section, we motivate the need for a multi-level OBDA approach, discussing scenarios in service management. We concentrate on the situation where the upper schema takes the form of a reference model, used by certain users to understand the business relationships existing between an organization and its external stakeholders. This is, e.g., the case of business managers and analysts, who often need to see the organization as an element of a bigger value chain, which consumes and provides services. To illustrate the challenges, we provide a concrete order scenario. In this scenario, the domain schema provides the basis to understand legacy data in terms of key concepts and relationships related to the notion of *order*. At the same time, managers employ the commitment-based core reference ontology for services (UFO-S) [20] as an upper schema, to understand and monitor the state of commitments that contractually relate the company and its customers. While the conceptual modeling community has made major advancements in the creation of reference models for services and commitments (as witnessed by UFO-S itself), no systematic approach to query organizational data through such reference models has been proposed so far. At the same time, recent attempts in formalizing relational, data-aware commitments, have either neglected the presence of legacy data [18] or the role of reference models [10].

2.1 The Order Scenario

In our order scenario, an organization called OHub acts as a hub between companies selling goods and persons interested in buying those goods. In particular, OHub takes care of an order-to-delivery process that supports a person in placing an order, allows her to pay for the order, and finally deliver the paid goods where the person desires. We assume that each order contains items that are all provided by the same company, but different orders may be about goods offered by different companies. Domain experts operating within OHub understand (some of) the key concepts and relations in this domain as specified by the UML class diagram of Figure 2, where the three subclasses of `Order` represent three distinct order states (for simplicity, we do not consider cancellation).

At the same time, the employees of OHub use a legacy management system to handle orders, and are unaware of how the actual data about orders and their involved stakeholders are stored. Such a management system relies on a relational database, whose schema is shown in Figure 3 together with some sample data. The schema consists of three tables, respectively keeping track of the stakeholders of OHub, of the orders managed by OHub, and of the respective money transfers. Specifically, the

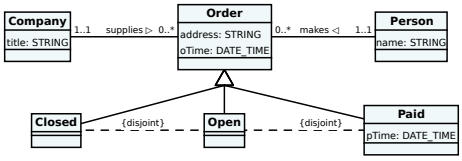


Fig. 2: UML class diagram of the order domain schema

type = p

type = c

final = 1

id	name	addr	type
pa	alice	bz-4	p
ce	eDvd	na-1	c
pb	bob	tn-3	p

Stakeholders

id	ctime	from	to	dest	final
o1	5	pa	ce	bz-5	0
o2	10	pb	ce	null	1
o3	20	pa	ce	null	1

OrderData

id	order	ttime
t1	o2	15

MTransfers

Fig. 3: Legacy schema of the order database, with some sample data. Directed edges denote foreign keys, contextualized to those tuples that satisfy the filter on the edge

Stakeholders table stores the main information about companies selling and delivering goods, and persons who buy such goods trough OHub. These two kinds of stakeholders are distinguished by the type attribute (which corresponds to string p for persons, and c for companies). The *OrderData* table stores the active and past orders managed by OHub. For each order, the table stores the creation time, the person who made it, the company responsible for the order, the destination address, and a flag indicating whether the order has been finalized by the person, or is still in the creation phase. It is intended that if the destination address is not specified (i.e., it is `null`), then the address of the person (as specified in the *Stakeholders* table) should be used for delivery. Finally, table *MTransfers* keeps track of monetary transfers related to finalized orders, for simplicity only recalling the transaction id, its time, and the order to which the transaction refers.

Managers of OHub often need to understand orders and their states in contractual terms, that is, as business objects and relationships that contractually bind OHub with its suppliers and customers. This is crucial, in turn, to assess the quality of service of OHub, and to detect whether the mutual commitments induced by order contracts are indeed honored or not. To do so, they adopt UFO-S as a reference model to define which services are offered by OHub, and which commitments relate to those services [20]. A portion of this reference model is shown in Figure 4. We only included the parts relevant for this small case study. We also incorporated explicit state attributes into the commitment classes, accounting for the different phases in which a commitment can be at a given time (pending, discharged, canceled, etc.) [10,18]. With UFO-S at hand, managers would like to inspect the negotiations and agreements established by

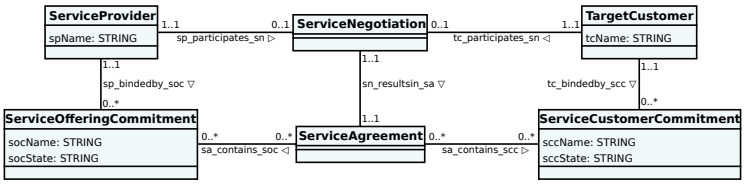


Fig. 4: Fragment of UFO-S used as upper schema extracted from UFO-S

companies and persons via OHub. We consider for simplicity only the following two commitments. Contractually, when a person makes an order, she becomes obliged to issuing a corresponding payment. In turn, upon payment the selling company becomes committed towards the actual delivery of the order.

2.2 Challenges

The OHub scenario presents a series of very interesting challenges in conceptual modeling and data access. Let us consider the need of OHub managers to inspect which commitments currently exist, and in which state they are. Obviously, they cannot directly formulate queries of this form on top of the legacy data, due to a vocabulary mismatch. A possible solution would be to create a dedicated OBDA specification that directly connects the legacy data to the UFO-S upper schema. However, this is unrealistic from the conceptual modeling point of view, for a twofold reason. First and foremost, linking data directly to concepts and relations in the upper schema requires to first understand the data in terms of domain notions, and only then to establish suitable connections between the domain and the upper level. For example, to inspect the contractual relationships between OHub's stakeholders, one has to clarify how orders, payments, and deliveries are reflected in the data, before identifying how mutual commitments about payments and deliveries are established and evolved. In addition, an OBDA specification connecting data to the domain schema could already be in place independently on these UFO-S related needs. It is well-known that creating an OBDA specification, especially for what concerns the understanding of the legacy data structures and the construction of corresponding mappings, is a labor-intensive and challenging task, similarly to alternative approaches to data access and integration [8]. If such a specification is already in place, it would be beneficial to build on it so as to gracefully integrate the upper schema into the picture, instead of creating another OBDA specification from scratch.

A second issue is related to the fact that reference models and upper ontologies are typically meant to capture a plethora of concepts and relations spanning over a wide range of application domains, with the purpose of resolving ambiguities and misunderstandings [20,22,13]. In a specific application domain such as that of OHub, only a small portion of the whole reference model is needed to capture the commitments of interest.

To attack these issues, we propose to ground the paradigm shown in Figure 1a into the concrete framework of Figure 1b. Here, a standard OBDA approach is adopted to make sense of the legacy data in terms of the domain schema. This can be used, e.g., to declare that each entry in the $OrderData$ table corresponds to an order, whose supplying company is obtained from the entry in $Stakeholders$ pointed by the to column, and whose making person is obtained from the entry in $Stakeholders$ pointed by the from column. The state of the order is, in turn, declared as follows: the order is *open* if the corresponding entry in $OrderData$ has $\mathsf{final} = 0$; *closed* if the corresponding entry in $OrderData$ has $\mathsf{final} = 1$, but no monetary transfer exists in $MTransfers$ for the order; *paid* if the corresponding entry in $OrderData$ has $\mathsf{final} = 1$, and there exists an entry in $MTransfers$ pointing to the order. Once this OBDA specification is in place, domain experts can forget about the schema of the legacy data, and work directly at the level of the domain schema. The domain schema is then employed to declare which concepts and relations define the UFO-S notions of service provider, target customer,

and corresponding offering and customer commitments. This identification step is seen as a declarative specification of how the domain schema of orders can be transformed into (a portion of) the UFO-S upper schema. Here, one could declaratively specify that: *(i)* Each closed order gives rise to a *pending* customer commitment binding its making person (i.e., its target customer) to paying it. *(ii)* Each paid order corresponds to a *discharged* customer commitment related to the order payment, and to a *pending* offering commitment binding its supplying company (i.e., its service provider) to delivering it.

Once the mapping and transformation rules are specified, OHub managers can express queries over UFO-S, and obtain answers automatically computed over the legacy data. For example, upon asking about all the pending commitments existing in the state of affairs captured by the data in Figure 3, one would get back two answers: one indicating that company `eDvd` has a pending commitment related to the delivery of order `o2`, and one telling that person `Alice` is committed to pay order `o3`.

This approach favors modularity and separation of concerns, since the mapping and the transformation rules can vary independently from each other. In particular, if the underlying data storage changes, only the mapping to the domain schema needs to be updated, without touching the definition of commitments. If instead the contract is updated, the domain-to-upper schema transformation needs to change accordingly, without touching the OBDA specification. In addition, the approach is driven by the actual querying requirements, that is, only the aspects of the upper schema that are relevant for querying need to be subject of transformation rules. The transformation rules themselves also provide a way to customize the view over the data. Even with the same upper schema, two different sets of transformation rules might provide different views over the data represented by the domain schema. We might even go beyond that, and consider situations where several upper schemas are provided, each with different sets of transformation rules. We do not explicitly consider this latter scenario in the following, since it can be obtained by multiple instantiations of the 2OBDA scenario.

3 Ontology-Based Data Access

We adopt a variant of Description Logics (DLs) [2] as a conceptual modeling language in which to express conceptual schemas. Specifically, we rely on *DL-Lite*$_\mathcal{A}$ [4], a prominent member of the *DL-Lite* family [5] of lightweight DLs, which have been specifically designed for efficiently accessing large amounts of data through an ontology. We introduce now such DL, and recall then the standard formalization of OBDA [21,4,23].

***DL-Lite*$_\mathcal{A}$ [21].** In *DL-Lite*$_\mathcal{A}$, the domain of interest is modeled through *concepts*, representing sets of (abstract) objects, *roles*, representing binary relations between objects, and *attributes*, representing binary relations between objects and values. Values belong to concrete datatypes, such as strings, integers, etc., but, for simplicity, we do not distinguish between different datatypes, and use a unique datatype instead (e.g., *STRING*) for all values in the system. The syntax of *DL-Lite*$_\mathcal{A}$ concept expressions B and role expressions R is given by the grammar

$$B \longrightarrow N \mid \exists R \mid \delta(U), \qquad\qquad R \longrightarrow P \mid P^-,$$

where N, P, and U (here and in the following) denote a *concept name*, a *role name*, and an *attribute name*, respectively. The role P^- denotes the *inverse of role* P, and

the concept $\exists R$, also called *unqualified existential restriction*, denotes the *domain* of a role R, i.e., the set of objects that R relates to some object. Similarly, the concept $\delta(U)$ denotes the *domain* of an attribute U, i.e., the set of objects that U relates to some value. Notice that the domain $\exists P^-$ of the inverse of role P actually represents the *range* of role P. Instead, since we have adopted a unique datatype for all values, there is no need to refer to the range of an attribute (since it is always, e.g., *STRING*).

A *DL-Lite*$_\mathcal{A}$ *Knowledge Base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a TBox $\mathcal{T}$, modeling intensional information, and an ABox $\mathcal{A}$, modeling extensional information.

The *TBox* consists of a finite set of *TBox assertions* of the following forms:

$$\begin{array}{lll} B_1 \sqsubseteq B_2, & B_1 \sqsubseteq \neg B_2, & \\ R_1 \sqsubseteq R_2, & R_1 \sqsubseteq \neg R_2, & (\mathsf{funct}\ R), \\ U_1 \sqsubseteq U_2, & U_1 \sqsubseteq \neg U_2, & (\mathsf{funct}\ U). \end{array}$$

From left to right, assertions of the first column respectively denote *inclusions* between concepts, roles, and attributes; assertions of the second column denote *disjointness* between concepts, roles, and attributes; assertions of the last column denote *functionality* of roles and attributes. In order to ensure the desired computational properties of *DL-Lite*$_\mathcal{A}$ that make it suitable as an ontology language in the context of OBDA, we impose that in a TBox, the roles and attributes occurring in functionality assertions cannot be specialized (i.e., they cannot occur in the right-hand side of inclusions) [5,4].

Example 1. We give some of the *DL-Lite* TBox assertions that capture the conceptual schema in Figure 2, where intuitively concepts correspond to classes, roles to binary associations, and DL attributes to UML attributes:

$$\begin{array}{llll} \mathit{Open} \sqsubseteq \mathit{Order}, & \mathit{Paid} \sqsubseteq \neg \mathit{Open}, & \exists \mathit{makes} \sqsubseteq \mathit{Person}, & \mathit{Order} \sqsubseteq \exists \mathit{makes}^-, \\ \mathit{Paid} \sqsubseteq \mathit{Order}, & \delta(\mathit{name}) \sqsubseteq \mathit{Person}, & \exists \mathit{makes}^- \sqsubseteq \mathit{Order}, & (\mathsf{funct}\ \mathit{makes}^-), \quad \ldots \end{array}$$

These assertions state that *Open* and *Paid* are sub-concepts of *Order* and that *Paid* and *Open* are disjoint. They further specify the domain of *name* and *makes* and the range of *makes*, that orders are made by someone, and that the inverse of *makes* is functional. ■

The *Abox* consists of a finite set of *ABox assertions* of the form $N(t_1)$, $P(t_1, t_2)$, or $U(t_1, v)$, where t_1 and t_2 denote *individuals* (representing abstract objects) and v denotes a *value*. A *constant* is an individual or a value. The *domain of an ABox* $\mathcal{A}$, denoted by $\textsc{adom}(\mathcal{A})$, is the (finite) set of constants appearing in $\mathcal{A}$.

For the semantics of *DL-Lite*$_\mathcal{A}$, we note that we interpret objects and values over disjoint domains, and that for both we adopt the *Unique Name Assumption*, i.e., different terms denote different objects (or values). The various types of TBox assertions (i.e., inclusion, disjointness, and functionality) are satisfied when the corresponding condition holds for the interpretation of the involved concepts, role(s), or attribute(s). We refer to [4] for details. We also adopt the usual notions of *satisfaction*, *model*, and *entailment* [4].

We point out that *DL-Lite*$_\mathcal{R}$, which is the sublanguage of *DL-Lite*$_\mathcal{A}$ obtained by dropping functionality assertions, is the formal counterpart of the OWL 2 QL language standardized by the W3C [19] as a profile of the Web Ontology Language OWL 2.

Queries. We consider queries that are formulated over the conceptual schema elements, and whose answers are formed by terms of the ABox. A *conjunctive query* (CQ) $q(\vec{x})$ over a TBox $\mathcal{T}$ is a first-order logic (FOL) formula of the form $\exists \vec{y}.conj(\vec{x}, \vec{y})$, whose

free variables (a.k.a. *answer variables*) are $\vec{x}$. The formula $conj(\vec{x}, \vec{y})$ is a conjunction of atoms of the form $N(z)$, $P(z, z')$, and $U(z, z')$ where N, P and U respectively denote a concept, a role, and an attribute name of $\mathcal{T}$, and z, z' are constants, or variables in $\vec{x}$ or $\vec{y}$. If q has no answer variables, then it is called *boolean*. A *union of conjunctive queries* (UCQ) $q(\vec{x})$ over $\mathcal{T}$ is a disjunction of CQs with the same answer variables $\vec{x}$, i.e., a FOL formula of the form: $\exists \vec{y}_1.conj_1(\vec{x}, \vec{y}_1) \vee \cdots \vee \exists \vec{y}_n.conj_n(\vec{x}, \vec{y}_n)$.

The *(certain) answers* to a (U)CQ $q(\vec{x})$ over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the set $\mathsf{cert}(q, \mathcal{K})$ of substitutions[6] σ of the answer variables $\vec{x}$ with constants in $\text{ADOM}(\mathcal{A})$, such that the closed FOL formula $q\sigma$ evaluates to true in every model of $\mathcal{K}$. The answer to a boolean query is either true (i.e., the empty substitution) or false (i.e., no substitution). Computing $\mathsf{cert}(q, \mathcal{K})$ of a UCQ q over a *DL-Lite*$_\mathcal{A}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is polynomial in the size of $\mathcal{T}$ and in AC^0 in the size of $\mathcal{A}$. This is a consequence of the fact that *DL-Lite*$_\mathcal{A}$ enjoys *FOL rewritability*, which in our setting means that for every UCQ q, one can compute $\mathsf{cert}(q, \mathcal{K})$ by evaluating the UCQ $\mathsf{rew}(q, \mathcal{T})$ over $\mathcal{A}$ considered as a database, where $\mathsf{rew}(q, \mathcal{T})$ is the so-called *perfect rewriting* of q w.r.t. $\mathcal{T}$ [5,4,23].

Ontology-Based Data Access (OBDA). An *OBDA specification* is a triple $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, where: *(i)* $\mathcal{R}$ is relational database schema, consisting of a finite set of relation schemas, possibly with constraints, *(ii)* $\mathcal{T}$ is a *domain schema*, formalized as a *DL-Lite*$_\mathcal{A}$ TBox; and *(iii)* $\mathcal{M}$ is an *OBDA mapping*, consisting of a set of mapping assertions. These specify how to extract data from the underlying database and how to use them to instantiate the elements of the domain schema. To denote instances of the domain schema, we consider a countably infinite set Λ of (uninterpreted) functions, and apply them to the values retrieved from the database. Specifically, each *mapping assertion* has the form $\varphi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$, where $\varphi(\vec{x})$ is an arbitrary SQL query over $\mathcal{R}$ with $\vec{x}$ as output variables, and $\psi(\vec{y}, \vec{t})$ is a conjunction of atoms over $\mathcal{T}$ using variables $\vec{y} \subseteq \vec{x}$ and terms $\vec{t}$, where each term in $\vec{t}$ has the form $f(\vec{z})$, with $f \in \Lambda$ and $\vec{z} \subseteq \vec{x}$.

Example 2. In our running example, the following mapping assertion is used to populate the concept *Person* and the corresponding attribute *name*, by selecting in the table *Stakeholders* entries for which the value of type equals 'p':

```
SELECT id as pid, name FROM Stakeholders WHERE type = 'p'
```
$$\rightsquigarrow \quad \mathit{Person}(\mathbf{person}(\texttt{pid})) \wedge \mathit{name}(\mathbf{person}(\texttt{pid}), \texttt{name})$$

The following mapping assertion is used to populate the role *makes* with all pairs consisting of an order and the corresponding person who made the order:

```
SELECT OD.id as oid, S.id as pid FROM OrderData OD, Stakeholders S
WHERE OD.from = S.id
```
$$\rightsquigarrow \quad \mathit{makes}(\mathbf{person}(\texttt{pid}), \mathbf{order}(\texttt{oid}))$$ ■

Given an OBDA specification $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$ and a database instance $\mathcal{D}$ that conforms to $\mathcal{R}$, the pair $\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$ is called an *OBDA instance* (for $\mathcal{S}$). Each mapping assertion $m = \varphi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$ in $\mathcal{M}$ generates from $\mathcal{D}$ a set $m(\mathcal{D}) = \bigcup_{\vec{c} \in \mathsf{eval}(\varphi, \mathcal{D})} \psi[\vec{x}/\vec{c}]$ of ABox assertions, where $\mathsf{eval}(\varphi, \mathcal{D})$ denotes the evaluation of $\varphi(\vec{x})$ over $\mathcal{D}$, and where $\psi[\vec{x}/\vec{c}]$ is the set (as opposed to the conjunction) of atoms obtained by substituting in $\psi(\vec{y}, \vec{t})$ each variable $x \in \vec{x}$ with the corresponding constant $c \in \vec{c}$. Then, the *(virtual) ABox*[7] *generated by* $\mathcal{M}$ *from* $\mathcal{D}$ is $\mathcal{M}(\mathcal{D}) = \bigcup_{m \in \mathcal{M}} m(\mathcal{D})$. Given an OBDA instance

[6] As customary, we view a substitution of $\vec{x}$ as a tuple of constants, one for each variable in $\vec{x}$.

[7] Such ABox is called *virtual* because in general it is not actually materialized.

$\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$ with $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, a *model* for $\mathcal{J}$ is a model of the KB $\langle \mathcal{T}, \mathcal{M}(\mathcal{D}) \rangle$. We say that $\mathcal{J}$ is *satisfiable* if it admits a model.

A UCQ q over an OBDA specification $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$ is simply a UCQ over $\mathcal{T}$, and given an OBDA instance $\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$, the *certain answers to q over $\mathcal{J}$*, denoted $\mathsf{cert}(q, \mathcal{J})$, are defined as $\mathsf{cert}(q, \langle \mathcal{T}, \mathcal{M}(\mathcal{D}) \rangle)$. To compute $\mathsf{cert}(q, \mathcal{J})$, instead of materializing the virtual ABox $\mathcal{M}(\mathcal{D})$, we can apply the following three-step approach [21]: *(i) rewrite* q to compile away $\mathcal{T}$, obtaining $q_r = \mathsf{rew}(q, \mathcal{T})$; *(ii)* use the mapping $\mathcal{M}$ to *unfold* q_r into a query over $\mathcal{R}$, denoted by $q_u = \mathsf{unf}(q_r, \mathcal{M})$, which turns out to be an SQL query; *(iii)* evaluate q_u over $\mathcal{D}$, obtaining $\mathsf{eval}(q_u, \mathcal{D})$. It has been shown in [21] that $\mathsf{cert}(q, \langle \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle, \mathcal{D} \rangle) = \mathsf{eval}(\mathsf{unf}(\mathsf{rew}(q, \mathcal{T}), \mathcal{M}), \mathcal{D})$.

Example 3. Recall our example, and consider the query $q(x) = \mathit{Person}(x)$ that retrieves all persons. Since $\mathcal{T}$ contains $\exists \mathit{makes} \sqsubseteq \mathit{Person}$ and $\exists \mathit{name} \sqsubseteq \mathit{Person}$, the rewriting of $q(x)$ w.r.t. $\mathcal{T}$ gives us the UCQ $q_r(x) = \mathit{Person}(x) \vee \exists y.\mathit{makes}(x, y) \vee \exists z.\mathit{name}(x, z)$, and the unfolding of $q_r(x)$ w.r.t. $\mathcal{M}$ gives us the following SQL query $q_u(x)$:

```
SELECT id as x FROM Stakeholders WHERE type = 'p'     UNION
SELECT S.id as x FROM OrderData OD, Stakeholders S WHERE OD.from = S.id
```
■

4 2-Level Ontology Based Data Access

Within an OBDA specification $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, the TBox $\mathcal{T}$ provides a specific conceptual view, which we called *domain schema*, of the domain of interest, through which the data sources $\mathcal{R}$ are accessed. However, as illustrated in Section 2, users might be interested in accessing the same data sources through a different conceptual schema $\mathcal{T}'$, which we called *upper schema*, without the need to redefine from scratch the relationship between $\mathcal{T}'$ and $\mathcal{R}$. Thus, we need a mechanism to relate the upper schema $\mathcal{T}'$ to the domain schema $\mathcal{T}$, and to exploit this connection, together with the information on $\mathcal{T}$ and $\mathcal{M}$ encoded in $\mathcal{S}$, for answering queries over the upper schema.

Here we introduce a solution based on transformation rules that declaratively map the domain schema $\mathcal{T}$ to the upper schema $\mathcal{T}'$ and that are based on so-called *GAV mappings*. Such form of mappings have been widely used in the data integration setting (cf. [16]) as a natural means to connect a global schema used for querying a data integration system to the schemas of the underlying data sources. Formally, given two TBoxes $\mathcal{T}$ and $\mathcal{T}'$, a *schema transformation* from $\mathcal{T}$ to $\mathcal{T}'$, is a set $\mathcal{N}$ of *transformation rules*, each of the form $\varphi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$, where $\varphi(\vec{x})$ is a UCQ over $\mathcal{T}$ with answer variables $\vec{x}$, and $\psi(\vec{y}, \vec{t})$ is a conjunction of atoms over $\mathcal{T}'$ over variables $\vec{y} \subseteq \vec{x}$ and terms $\vec{t}$. Similarly to the case of OBDA mappings, the terms in $\vec{t}$ have the form $f(\vec{z})$, with $f \in \Lambda$ and $\vec{z} \subseteq \vec{x}$, and they are used to construct the identifiers of individuals for $\mathcal{T}'$. However, such identifiers might also be returned directly by the query φ. Essentially, such a schema transformation populates the elements of the upper schema $\mathcal{T}'$ with the information obtained from the answers to queries over the domain schema $\mathcal{T}$. Formally, given an ABox $\mathcal{A}$ for $\mathcal{T}$, a transformation rule $r = \varphi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$ in $\mathcal{N}$ generates a set $r(\mathcal{T}, \mathcal{A}) = \bigcup_{\vec{c} \in \mathsf{cert}(\varphi, \langle \mathcal{T}, \mathcal{A} \rangle)} \psi[\vec{x}/\vec{c}]$ of ABox assertions for $\mathcal{T}'$, where $\psi[\vec{x}/\vec{c}]$ is defined as for OBDA mapping assertions. Notice that since a transformation rule is applied to a KB (and not to a database instance), the query in the left-hand side of the

rule is interpreted under certain-answer semantics. Then, the ABox generated by $\mathcal{N}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ is defined as $\mathcal{N}(\mathcal{T}, \mathcal{A}) = \bigcup_{r \in \mathcal{N}} r(\mathcal{T}, \mathcal{A})$.

Example 4. The transformation rule $\mathit{Person}(x) \rightsquigarrow \mathit{TargetCustomer}(\mathbf{tc}(x))$ maps each instance of the domain schema concept *Person* into the upper schema concept *TargetCustomer*. ■

We are now ready to formalize the setting of *2-level OBDA*, or *2OBDA* for short. A *2OBDA specification* is a tuple $\mathcal{S}_{\langle 2 \rangle} = \langle \mathcal{S}, \mathcal{N}, \mathcal{T}' \rangle$, where $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$ is an OBDA specification, $\mathcal{T}'$ is an upper schema (expressed as a *DL-Lite*$_\mathcal{A}$ TBox), and $\mathcal{N}$ is a schema transformation from $\mathcal{T}$ to $\mathcal{T}'$. Given an OBDA instance $\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$ for $\mathcal{S}$, the tuple $\mathcal{J}_{\langle 2 \rangle} = \langle \mathcal{J}, \mathcal{N}, \mathcal{T}' \rangle$ is called a *2OBDA instance* for $\mathcal{S}_{\langle 2 \rangle}$. The notions of virtual ABox and query answering in OBDA can be lifted to 2OBDA as well. Let $\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$, with $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, be a satisfiable OBDA instance. The (virtual) ABox $\mathcal{N}(\mathcal{J})$ generated by $\mathcal{N}$ from $\mathcal{J}$ is defined as $\mathcal{N}(\mathcal{J}) = \mathcal{N}(\mathcal{T}, \mathcal{M}(\mathcal{D}))$. Moreover, a *model* for $\mathcal{J}_{\langle 2 \rangle}$ is a model of the KB $\langle \mathcal{T}', \mathcal{N}(\mathcal{J}) \rangle$, and $\mathcal{J}_{\langle 2 \rangle}$ is *satisfiable* if it admits a model. A UCQ q over a 2OBDA instance $\mathcal{J}_{\langle 2 \rangle}$ is simply a UCQ over the upper schema $\mathcal{T}'$, and the *certain answers to q over $\mathcal{J}_{\langle 2 \rangle}$*, denoted $\mathsf{cert}(q, \mathcal{J}_{\langle 2 \rangle})$, are defined as $\mathsf{cert}(q, \langle \mathcal{T}', \mathcal{N}(\mathcal{J}) \rangle)$.

To compute $\mathsf{cert}(q, \mathcal{J}_{\langle 2 \rangle})$, instead of materializing the intermediate and final virtual ABoxes, we can use the following 5-step approach: *(i) rewrite* q to compile away the upper schema $\mathcal{T}'$, obtaining $q'_r = \mathsf{rew}(q, \mathcal{T}')$, which is a UCQ over $\mathcal{T}'$; *(ii)* use the schema transformation $\mathcal{N}$ to *unfold* q'_r into a query over the domain schema $\mathcal{T}$, denoted by $q'_u = \mathsf{unf}(q'_r, \mathcal{N})$, which turns out to be a UCQ; *(iii) rewrite* q'_u to compile away the domain schema $\mathcal{T}$, obtaining $q_r = \mathsf{rew}(q'_u, \mathcal{T})$; *(iv)* use the mapping $\mathcal{M}$ to *unfold* q_r into a query over $\mathcal{R}$, denoted by $q_u = \mathsf{unf}(q_r, \mathcal{M})$, which turns out to be an SQL query; *(v)* evaluate q_u over $\mathcal{D}$, obtaining $\mathsf{eval}(q_u, \mathcal{D})$. The correctness of this approach for computing the certain answers $\mathsf{cert}(q, \mathcal{J}_{\langle 2 \rangle})$ can be shown by considering that essentially it combines two rewriting and unfolding phases for computing certain answers, and that both combinations are correct since they are based on standard OBDA [21]. We get:

Theorem 1. *Given a 2OBDA instance $\mathcal{J}_{\langle 2 \rangle} = \langle \mathcal{J}, \mathcal{N}, \mathcal{T}' \rangle$, where $\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$ and $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, and a UCQ q over $\mathcal{T}$, we have that*

$$\mathsf{cert}(q, \mathcal{J}_{\langle 2 \rangle}) \;=\; \mathsf{eval}(\mathsf{unf}(\mathsf{rew}(\mathsf{unf}(\mathsf{rew}(q, \mathcal{T}'), \mathcal{N}), \mathcal{T}), \mathcal{M}), \mathcal{D}).$$

This result provides a way for answering queries in 2OBDA, but at the cost of an ah-hoc solution for query translation. We present now an alternative approach based on deriving from a 2OBDA specification an equivalent standard OBDA specification. Specifically, we show how to compile away from a 2OBDA specification $\mathcal{S}_{\langle 2 \rangle} = \langle \mathcal{S}, \mathcal{N}, \mathcal{T}' \rangle$, where $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, the domain schema $\mathcal{T}$, and synthesize from $\mathcal{S}_{\langle 2 \rangle}$ a standard OBDA specification $\mathcal{S}' = \langle \mathcal{R}, \mathcal{M}', \mathcal{T}' \rangle$ in which the elements of the upper schema $\mathcal{T}'$ are directly mapped to the sources. To construct the OBDA mapping $\mathcal{M}'$ of $\mathcal{S}'$, we first rewrite w.r.t. $\mathcal{T}$ and then unfold w.r.t. $\mathcal{M}$ each query over the domain schema in each transformation rule in $\mathcal{N}$. Formally, for each transformation rule $\varphi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$ in $\mathcal{N}$, the mapping $\mathcal{M}'$ contains a mapping assertion $\varphi'(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$, where $\varphi' = \mathsf{unf}(\mathsf{rew}(\varphi, \mathcal{T}), \mathcal{M})$.

Example 5. Continuing Example 4, by applying the steps above, we get the OBDA mapping $q_u(x) \rightsquigarrow \mathit{TargetCustomer}(\mathbf{tc}(x))$, where $q_u(x)$ is the SQL query in Example 3. ■

To show the correctness of this approach, consider a source database $\mathcal{D}$ conforming to $\mathcal{R}$, and the corresponding OBDA instance $\mathcal{J} = \langle \mathcal{S}, \mathcal{D} \rangle$ and 2OBDA instance

$\mathcal{J}_{\langle 2\rangle} = \langle \mathcal{J}, \mathcal{N}, \mathcal{T}' \rangle$. By the correctness of the rewriting and unfolding approach to OBDA (cf. [21]), for each transformation rule $r = \varphi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$ in $\mathcal{N}$, we have that $\mathsf{cert}(\varphi, \mathcal{J}) = \mathsf{cert}(\varphi, \langle \mathcal{T}, \mathcal{M}(\mathcal{D})\rangle) = \mathsf{eval}(\varphi', \mathcal{D})$, where $\varphi' = \mathsf{unf}(\mathsf{rew}(\varphi, \mathcal{T}), \mathcal{M})$. Hence, $\varphi(\vec{x})$ in r and $\varphi'(\vec{x})$ in the mapping assertion m of $\mathcal{M}'$ derived from r return the same answers, and in both r and m these instantiate the same atoms in $\psi(\vec{y}, \vec{t})$. It follows that $\mathcal{N}$ and $\mathcal{M}'$ populate the elements of the TBox $\mathcal{T}'$ with the same facts. We get:

Theorem 2. *Let $\mathcal{S}_{\langle 2\rangle} = \langle \mathcal{S}, \mathcal{N}, \mathcal{T}' \rangle$ be a 2OBDA specification, where $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, and let $\mathcal{S}' = \langle \mathcal{R}, \mathcal{M}', \mathcal{T}' \rangle$ be the OBDA specification derived from $\mathcal{S}_{\langle 2\rangle}$ as specified above. Then $\mathcal{S}_{\langle 2\rangle}$ and $\mathcal{S}'$ have the same models.*

By relying on this result, query answering over a 2OBDA instance can be delegated to a traditional OBDA system, and we can use for it existing implementations (e.g., [3]).

5 Using Annotations for Specifying Schema Transformations

To specify the schema transformation rules between the domain ontology and the upper ontology, we propose an approach based on annotations, which are then used to actually generate the rules. We employ UML class diagrams as a concrete language for conceptual data modeling, and we rely on their logic-based encoding in terms of OWL 2 QL [4,6]. Therefore, we assume to work with OWL 2 QL compliant ontologies. The available types of annotations are automatically deduced from the upper ontology based on this assumption. In fact, we have developed an editor for annotating the domain ontology with upper ontology concepts that dynamically builds the annotation types accordingly.

Every class/concept in the ontology is considered to be an annotation type. Data properties having this class as domain are considered as fields of this annotation and they will be populated from the fields available in the domain ontology. Object properties are instead populated in their range class. It is possible to select any data property of the domain ontology or enter a static value to the data properties. For object properties, it is necessary to select an annotation instance defined over the domain ontology. For both data and object properties, the user can exploit navigational access, which provides the knowledge of how to access any field or annotation over the UML diagram using relations between classes as paths. After the selection of the corresponding fields, values, or navigational paths for each element in the annotation, queries for the transformation rules are automatically generated and actually used for rule generation (cf. Section 4).

The different annotations enrich the domain schema $\mathcal{S}$ by indicating which classes/concepts, associations/roles, and attributes in $\mathcal{S}$ contribute to the identification of different attributes of the annotations extracted from $\mathcal{T}'$. Towards the automated processing of such annotations, and the consequent automated generation of transformation rules, the first step is to formally represent the annotations using a machine-processable language. To this end, we rely on conjunctive queries encoded in SPARQL, which are used to extract objects/values targeted by the annotations. In this way, we can represent GAV mappings that go well beyond one-to-one correspondences between classes/associations.

We provide a tool-chain that supports the various phases of the 2OBDA design and in particular implements the automated processing technique for annotations. It is available as a stand-alone software, or as a set of plugins running inside the ProM process mining

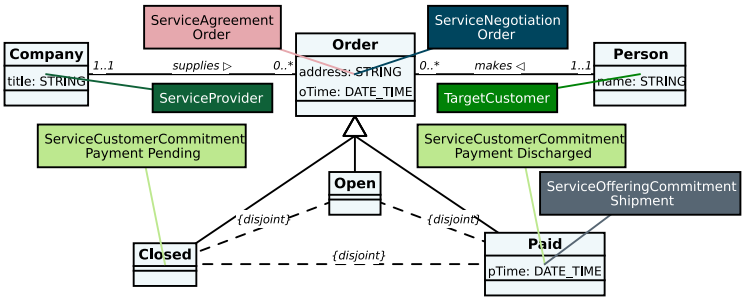
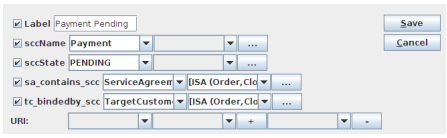


Fig. 5: Annotating the domain ontology using concepts from the upper ontology



(a) Payment pending service customer comm.

(b) Payment discharged service customer comm.

(c) Shipment service offering commitment

Fig. 6: Forms of commitment annotations showing the values

framework[8]. Specifically, it consists of the following components: *(i)* A *UML Editor*, to model the domain and upper ontologies as UML class diagrams, and to import from and export to OWL 2 QL. *(ii)* A *Dynamic Annotation Editor*, to enrich the domain ontology with annotations extracted from the upper ontology. The editor supports navigational selections over the diagram via mouse-click operations to simplify the annotation task. The annotations are automatically translated into corresponding SPARQL queries by the editor. *(iii)* A *Transformation Rule Generator*, which automatically processes the annotations, and generates rules between the domain and upper ontologies. It implements the mapping synthesis technique described in Section 4, leveraging the state-of-the-art framework *ontop*[9] [3] for mapping management and query rewriting and unfolding.

Currently we do not have native tool support for specifying the mapping between the domain ontology and the data sources, and we assume that it is realized manually or by exploiting third-party tools, such as the *mapping editor* in the *ontop* plugin for Protégé.[9]

To use the Dynamic Annotation Editor to annotate the domain ontology with the concepts of the upper ontology, a user first defines the upper ontology as the target ontology, and then loads the file with the domain ontology to annotate. For example, if we use the partial UFO-S ontology as our upper ontology, and use it to annotate the order ontology, we can easily extract commitments available in the system. Figure 5

[8] `http://onprom.inf.unibz.it`
[9] `http://ontop.inf.unibz.it`

shows example annotations for extracting the commitments required by the company from the order system and Figure 6 shows the forms of corresponding annotations. Note that Figure 2 and Figure 4 are image exports taken from the UML Editor, and Figure 5 is an image export taken from the Dynamic Annotation editor.

According to the annotations defined in Figure 5, the `TargetCustomer`s are extracted from the `Person` concept and `ServiceProvider`s from the `Company`. The `Order` concept defines a `ServiceNegotiation` and a `ServiceAgreement` between a `Company` and a `Person`. Two `ServiceCustomerCommitment`s for the `Payment` are extracted when a `Person` gives an `Order` over the `Closed` (for pending payment commitment) and `Paid` (for discharged payment commitment) specializations of `Order`. After the `Payment` is successfully discharged, a `ServiceOfferingCommitment` for the `Shipment` will be available in the system. Figure 5 depicts the commitment extraction requirements of the company regarding their order system, and their presentation in the form of a diagram allows users to understand the relations and the requirements in a more intuitive way.

The tool-chain provides an easy annotation mechanism to the users with diagrams, forms, and point-and-click actions, so they don't have to deal with any text processing or ETL kind of approaches. The mapping generator of the tool-chain automatically generates the mapping between the underlying data sources and the upper ontology, using the domain ontology, the mapping specification between data sources and domain ontology, the upper ontology, and the annotations defined over the domain ontology.

6 Conclusions

In this paper, we proposed a framework for accessing data through different conceptual schemas, e.g., a domain schema and an upper schema acting as a reference model for the domain. We have formalized the framework in terms of 2-level OBDA, and have shown how to exploit an existing OBDA specification for the domain schema, together with conceptual mappings between the domain and the upper schema, to automatically derive an new OBDA specification for the upper schema. We have shown how the framework can be realized through schema annotations, and we have implemented a tool-chain supporting such approach. We have illustrated our approach on a use case adopting as upper schema a commitment-based core reference ontology for service (UFO-S).

Another interesting and highly relevant setting that calls for a multi-level approach to data access, is that of *process mining*, sketched in Figure 1c [1]. In [8,6,7], a framework for extracting event logs using OBDA has been proposed, considering the XES standard for event logs [14] as a fixed, upper schema that is hard-coded in the framework. The 2OBDA approach presented here allows us to overcome the rigidity of XES and use instead a customized upper schema that can account e.g., for events with start and end time, or for hierarchical process instances. Finally, we observe that the framework of 2-level OBDA and the results in Section 4, can be easily generalized to *multiple-levels*, where schema transformations are specified between multiple conceptual schemas.

Acknowledgements. This research is supported by the Euregio IPN12 *KAOS (Knowledge-Aware Operational Support)* project, funded by the "European Region Tyrol-South Tyrol-Trentino" (EGTC), and by the UNIBZ internal project *OnProm (ONtology-driven PROcess Mining)*.

References

1. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Proc. of the BPM Int. Workshops. LNBIP, vol. 99, pp. 169–194. Springer (2012)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. CUP (2003)
3. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web J. **8**(3), 471–487 (2017)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: The *DL-Lite* approach. In: Tessaris, S., Franconi, E. (eds.) RW Tutorial Lectures, LNCS, vol. 5689, pp. 255–356. Springer (2009)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. JAR **39**(3) (2007)
6. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A.: OBDA for log extraction in process mining. In: RW Tutorial Lectures, LNCS, vol. 10370, pp. 292–345. Springer (2017)
7. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A.: The onprom toolchain for extracting business process logs using ontology-based data access. In: Proc. of the BPM Demo Track and BPM Dissertation Award, co-located with BPM 2017. CEUR, vol. 1920 (2017)
8. Calvanese, D., Kalayci, T.E., Montali, M., Tinella, S.: Ontology-based data access for extracting event logs from legacy data: The onprom tool and methodology. In: Proc. of BIS. LNBIP, vol. 288, pp. 220–236. Springer (2017)
9. Catarci, T., Lenzerini, M.: Representing and using interschema knowledge in cooperative information systems. JICIS **2**(4), 375–398 (1993)
10. Chopra, A.K., Singh, M.P.: Custard: Computing norm states over information stores. In: Proc. of AAMAS. pp. 1096–1105 (2016)
11. Daraio, C., et al.: The advantages of an ontology-based data management approach: Openness, interoperability and data quality. Scientometrics **108**(1), 441–455 (2016)
12. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, 2nd edn. (2013)
13. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In: Proc. of DB&IS. pp. 18–39. IOS Press (2006)
14. IEEE Computational Intelligence Society: IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. Std 1849-2016, IEEE (2016)
15. Kharlamov, E., et al.: Ontology based data access in Statoil. J. of Web Semantics **44** (2017)
16. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of PODS (2002)
17. Mehdi, G., et al.: Semantic rule-based equipment diagnostics. In: Proc. of ISWC. LNCS, vol. 10588, pp. 314–333. Springer (2017)
18. Montali, M., Calvanese, D., De Giacomo, G.: Verification of data-aware commitment-based multiagent systems. In: Proc. of AAMAS. pp. 157–164 (2014)
19. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, W3C (Dec 2012)
20. Nardi, J.C., de Almeida Falbo, R., Almeida, J.P.A., Guizzardi, G., Pires, L.F., van Sinderen, M.J., Guarino, N., Fonseca, C.M.: A commitment-based reference ontology for services. Information Systems **54**, 263 – 288 (2015)
21. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics **X**, 133–173 (2008)
22. Scherp, A., Saathoff, C., Franz, T., Staab, S.: Designing core ontologies. Applied Ontology **6**(3), 177–221 (2011)
23. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: Proc. of IJCAI. AAAI Press (2018)