Improving the Data Quality of Relational Databases using OBDA and OWL 2 QL

Olivier Curé

Université Paris-Est, IGM Terre Digitale, Marne-la-Vallée, France
olivier.cure@univ-paris-est.fr.fr

1 Introduction

The Semantic Web aims to enable the integration and sharing of data across different applications and organizations. A first step toward reaching this goal is a closer cooperation between the best technology to deal with large volumes of data, i.e. currently relational DBMS, and ontologies. The Ontology-Based Data Access (OBDA) approach [7] tackles this issue by providing a conceptual view over data repositories and inference enabled query answering solutions. OBDA is based on the *DL–Lite* family of Description Logics (DL) [2] which have inspired the OWL 2 QL profile.

In this paper, we consider a novel aspect of OBDA systems: their reasoning services and query answering solutions could be used to improve the data quality of the source databases. This can be performed by checking the satisfaction of a set of constraints over database instances. Although many attempts to add constraints to OWL ontologies have been proposed, few of them apply to the OBDA context.

In this work, we consider novel data dependencies that have recently been introduced in the database domain [5]: Conditional Dependencies (CDs). They correspond to Conditional Functional Dependencies (CFDs) and Conditional Inclusion Dependencies (CINDs) and are conditional counter part of respectively functional and inclusion dependencies (henceforth FDs and INDs). Intuitively, CDs hold only for the tuples that satisfy some conditions and are supposed to capture more of the inconsistencies of real-life data. We claim that inferences performed in the context of OBDA provide a more compact, and thus efficient, representation of these CDs. Two main problems associated to CDs are their discovery and their representation.

In the context of relational DBMS, several techniques have recently been proposed to discover ([6] for CFDs and [4] for CINDs) and represent (via SQL queries) these CDs. Concerning discovery, we claim that it is more efficient to discover CDs directly from the database sources and thus enjoying existing optimized implementations. Concerning representation, since OWL 2 QL does not support the introduction of individual names in concept descriptions, we also adopted a query representation. Since most OBDA systems do not support negation as failure patterns in SPARQL (i.e. through the introduction of negation and bound operators), we have selected the SparSQL epistemic query language

[3]. Thus CDs are provided by an external solution and we need to translate them in SparSQL. In order to obtain a minimal set of these queries, this translation exploits standard DL reasoning services, i.e. concept subsumption.

2 Conditional Dependencies

In this section, we present the syntax and semantics of CFDs and CINDs which are respectively extension of standard FDs and INDs with patterns.

A CFD ψ defined on a relation R is a pair $R(X \rightarrow Y, T_p)$ where X and Y are attribute sets in R, $X \rightarrow Y$ is a standard FD [1] and T_p is a pattern tableau containing all attributes in X and Y. The sets X and Y are of the same arity and for each attribute A in $(X \cup Y)$, the value of the attribute A of the tuple t_p in T_p, denoted $t_p[A]$ is either a value from the domain of A or a variable denoted by '_'. A tuple t matches a tuple t_p in T_p if for each attribute A in T_p, for some constant 'a', $t[A] = t_p[A]$ ='a' or $t_p[A]$ ='_'.

An instance D of R satifies the CFD ψ, denoted $D \models \psi$, if for every pair of tuples t_1 and t_2 in D, and for each pattern tuple t_p in T_p, if $t_1[A] = t_2[A]$ for every attribute A in X, and both t_1 and t_2 match $t_p[A]$, then $t_1[B] = t_2[B]$ and t_1, t_2 both match $t_p[B]$ for each attribute B in Y.

A CIND ϕ defined over a pair of relations R_1 and R_2 is a pair $(R_1(X; X_p) \subseteq R_2(Y; Y_p), T_p)$ where X, X_p and Y, Y_p are attribute sets of R_1 and R_2 respectively, $R_1(X) \subseteq R_2(Y)$ is a standard IND [1] and T_p is a pattern tableau of ϕ with attribute sets X_p and Y_p such that each pattern tuple t_p and each attribute B in X_p (or Y_p), $t_p[B]$ is a constant in the domain of B.

An instance (D_1, D_2) of (R_1, R_2) satisfies the CIND ϕ, denoted $(D_1, D_2) \models \phi$, iff for each tuple t_p in T_p and for each t_1 in D_1, if $t_1[X_p] = t_p[X_p]$, then there must exist t_2 in D_2 such that $t_1[X] = t_2[Y]$ and $t_2[Y_p] = t_p[Y_p]$.

Example 1: To illustrate these CDs, let us consider the following relational schema for book, cd and dvd data: dvd(iddvd, artist, title, type ,label, region), cd(idcd, artist, title, type ,label) and book(isbn, author, title, format, editor). In the following CDs, we represent pattern tableaux as sets of tuples where the attributes sets X and Y of a CFD (resp. $(X; X_p)$ and $(Y; Y_p)$ of a CIND) are delimited by the symbol '$\|$':
$\psi_1 = cd(label \rightarrow type, \{('bluenote' \| 'jazz'), ('DeutscheGramophon' \| 'classic')\}$
$\phi_1 = cd(title, label; type) \subseteq book(title, author; format), \{'\_', \_', 'audiobook' \|$
$'\_', \_', 'audio'\}$

Here, the first tuple in ψ_1's T_1 asserts that all *cd* tuples that have a 'Blue note' label must have a 'jazz' type. Interestingly, this condition also applies to the *dvd* relation, i.e. all Blue Note dvds have a 'jazz' type. ϕ_1 states that for each tuple in the *cd* relation with an 'audiobook' type, there must exits a book tuple with the same artist and title values and an 'audio' format.

3 System Description

Our OBDA system corresponds to a triple $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ where $\mathcal{T}$ is a TBox formalized in OWL 2 QL, $\mathcal{D}$ is relational database and $\mathcal{M}$ is a set of mapping assertions between $\mathcal{T}$ and $\mathcal{D}$. The mapping assertions support the extraction of the data from $\mathcal{D}$ to populate an ABox associated to the TBox $\mathcal{T}$. They take the form of Global Local As View (GLAV) mappings where the left handside of the assertion is a SQL query and the right handside is a conjunctive query over ontology entities. In the following mapping example, the *getcd* identification function enables to associate data stored in the relation to objects in the ontology:

SELECT idcd, artist,title, type, label FROM cd	$\rightsquigarrow$ Cd(getcd($idcd)),title(getcd($idcd),$title), artist(getcd($idcd),$artist), type(getcd($idcd),$type), label(getcd($idcd),$label)

In our running example, the TBox $\mathcal{T}$ corresponds to:

$AudioDocument \sqsubseteq Document$	$Dvd \sqsubseteq AudioDocument$
$NonAudioDocument \sqsubseteq Document$	$Cd \sqsubseteq AudioDocument$
$NonAudioDocument \sqcap AudioDocument \sqsubseteq \bot$	$Book \sqsubseteq NonAudioDocument$

Since the existential quantification to an individual or a literal and the enumeration of individuals or literals are not supported in OWL 2 QL, we need to represent the CDs with queries supported in OBDA. With these queries, we are not only interested in checking the satisfiability of a knowledge w.r.t. a set of CDs but we would also like to identify those tuples that do not satisfy a given CD. Thereby a form of negation is required in the query language. This is not compatible with the following OBDA restriction: queries posed over the ontology must correspond to union of conjunctive queries. This is motivated by maintaining LOGSPACE data complexity of query answering in OBDA and allows delegating reasoning on data to a RDBMS. Nevertheless, in [3] the authors propose an epistemic query language called *SparSQL* which satisfies our needs.

Another issue concerns CD discovery. Should it be discovered from the RDBMS or from the ontology? Since the generation of the pattern tableaux imposes to mine a database instance, we consider that it is more efficient to directly search for CDs at the RDBMS level. This is mainly due to the query answering process of OBDA which requires several steps, i.e. query expansion, unfolding and evaluation, and which would make a ontology-based discovery less computationally efficient. Additionally, complete, sound and optimized discovery algorithms are available for both CFDs [6] and CINDs[4] in a relational context. Thus an external solution provides sets of CFDs and CINDs and we need to translated them into *SparSQL* queries. A desirable feature of this translation is to compact these CDs using the DL concept subsumption reasoning service.

Our translation technique works as follows: for each CFD (resp. CIND) we use the mappings to search for the property mapped to each attribute in X and Y (resp. X, X_p, Y, Y_p for CINDs). This is performed by selecting the mapping assertions whose SQL queries contain these attributes as distinguished variables, hence they must be mapped to ontology elements. Then we search for the concepts associated to these properties using the identification functions, e.g. *getcd*.

Given this concept/relation binding, we can compact CDs that have common attribute sets and then search for the most common specific super concept. This is performed by assuming a covering constraint over subconcepts and using a standard DL reasoner. A straightforward generation of *SparSQL* queries terminates the transalation. For instance, the pattern tableau ψ_1 can be generalized, given our TBox, to both the *Cd* and *Dvd* concepts which are subsumed by *AudioDocument*. Thus we can generate the following *SparSQL* query which identifies *cd* and *dvd* tuples that violate this tableau:

```
SELECT q1.item FROM sparqltable ( SELECT ?item ?type ?label WHERE
{?item rdf:type 'AudioDocument'. ?item :typ ?type. ?item :label ?label}) q1
WHERE q1.label LIKE 'Blue note' and q1.type NOT LIKE 'jazz'
```

Similarly, the following query identifies *book* tuples violating ϕ_1:

```
SELECT q1.book FROM sparqltable ( SELECT ?book ?format
WHERE { ?book rdf:type 'Book'. ?item rdf:type 'AudioDocument'.
?item :label ?label. ?book :editor ?label. ?item :typ 'audio book'.
?book :format ?format.}) q1 WHERE q1.format NOT LIKE 'audio'
```

4 Discussion

This paper proposes a first approach to improve the data quality of relational databases via the introduction of conditional dependencies in an OBDA system. A prototype implementation has been developed using *QuOnto*, *Mastro–I* and recording the *SparSQL* queries in *Protégé4OBDA* plugin. It consists of an 'on-demand' solution which due to compactness of CDs enables an efficient identification of violating tuples and an easier maintenance of these dependencies by domain experts. We consider that this is not ideal and we are working on a trigger-based solution that would enable the processing of the right *SparSQL* query when a given relation is updated.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
3. C. Corona, E. D. Pasquale, A. Poggi, M. Ruzzi, and D. F. Savo. When dl-lite met owl.... In C. Dolbear, A. Ruttenberg, and U. Sattler, editors, *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
4. O. Curé. Conditional inclusion dependencies for data cleansing: Discovery and violation detection issues. In *QDB'09, To appear*.
5. W. Fan. Dependencies revisited for improving data quality. In *PODS*, pages 159–170, 2008.
6. W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *ICDE*, pages 1231–1234, 2009.
7. A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.