Incremental Generation of Mappings in an Ontology-Based Data Access Context

Olivier Curé

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

Abstract. Ontology-Based Data Access provides a conceptual view over data repositories and mediates the access to this information. The cornerstone of this approach consists of a set of mappings which express relationships between repository entities and ontology elements. In practice, these mappings may incorporate constant values. We propose a (semi) automatic solution that derives new mappings by analyzing the intensional and extensional levels of the sources as well as previously defined mappings containing constant values. Our solution also generates concept and role labels associated to the newly proposed mappings and proposes a graphical user interface for their adoption/rejection.

1 Introduction

The Semantic Web aims to enable the integration and sharing of data across different applications and organizations. This vision requires a close cooperation between relational databases (RDB) and ontologies. [7] presents the Ontology-Based Data Access (OBDA) approach which enables to represent an application domain's intensional level by an ontology. This ontology is then used for query purposes by exploiting reasoning services. Considering that the data level of this approach is managed by a RDBMS, a translation from queries expressed over the ontology to SQL queries is required.

The notion of mapping is a central aspect in the OBDA approach since they link relations of the DB to elements of the ontology. The standard OBDA architecture can be considered as a data integration system where the target corresponds to an ontology represented in a Description Logics (DL) formalism [2]. Current OBDA implementations require that the ontology is provided before mapping definitions. In this work, we assume an integration approach where the introduction of new concepts and roles in the mappings is allowed. Once these mappings are validated, new concepts and roles are integrated in the ontology and the end-user has the possibility to add axioms to them, e.g. subconcept relationship or domain/range definition for a role.

Based on this approach, the main contribution of this work is to propose a mapping generation solution. From our experience in mapping RDBs to ontologies, e.g. in ecological domain [5], we have noticed that constant values are frequently required when the sources contain terminologies and one needs to define fine-grained ontologies. Without the help of a semi-automatic solution, the

R. Meersman, T. Dillon, P. Herrero (Eds.): OTM 2009, Part II, LNCS 5871, pp. 1025–1032, 2009.

end-user would have to define all mappings and ontology elements manually. An error-prone and time consuming task. Our solution takes one end-user generated mapping containing constants from DB instances as an input and generates similar mappings based on the discovery of uncovered DB constants. Moreover, if concepts or roles are introduced in the input mapping, then the system searches, using background knowledge, for appropriate concept or role labels to introduce in the generated mappings. Since some of these mappings may not satisfy the end-user, we propose a solution for their validation and refinement.

This paper is organized as follows: Section 2 presents basic notions. Section 3 presents the mapping generation solution. Section 4 deals with finding concept/role labels and proposes a refinement/validation solution. Section 5 proposes some related work and Section 6 concludes the paper.

2 Preliminaries

A database schema is a finite sequence R=$\{R_1, .., R_n\}$ of relation symbols R_i, with $1 \leq i \leq n$, that are of fixed arity and correspond to the DB relations. We define a relation R_i as a set of attributes $\{A_i^1, .., A_i^k\}$ where A_i^j, with $1 \leq j \leq k$, denotes an attribute over a possibly infinite domain Δ.

Example 1: Consider a DB with data about drugs. It contains the relations **drug** with attributes *cip* (a drug identifier), *name* and *labId*, **labs** with attributes *labId* (laboratory identifier) and *labName*, **therapeutics** with attributes *classId* and *className*. Finally the relation **drugTherapeutics** relates drugs to their therapeutic classes using attributes *cip* and *classId*.

We consider that ontologies are expressed in $DL-Lite_A$ [7]. This DL has sufficient expressive power to capture the conceptual languages used in DB modeling, e.g. Entity Relationship (ER) model and Unified Modeling Language (UML) class diagram. The main features of $DL-Lite_A$ correspond to 'is-a' and disjointness between classes and properties, domain and range of properties, mandatory participation and functionality of relations. We assume readers are familiar with the semantics of DL [2].

Example 2: Let $\mathcal{T}$ be an extract of a $DL-Lite_A$ medical ontology:

$Drug \sqsubseteq \top_C$ $\quad$ $funct(drugName)$ $\quad$ $Drug \sqsubseteq \delta(drugName)$

$Lab \sqsubseteq \top_C$ $\quad$ $\rho(drugName) \sqsubseteq xsd : string$ $\quad$ $\exists producedBy^- \sqsubseteq Lab$

$Lab \sqsubseteq \neg Drug$ $\quad$ $Drug \sqsubseteq \exists producedBy$

OBDA uses Global Local As View (GLAV) mappings which map a query over the sources to a conjunctive query over elements of the target. Formally, these mappings take the following form: $\Phi(\overrightarrow{x}) \rightsquigarrow \Psi(f(\overrightarrow{x}), \overrightarrow{x})$, where $\Phi(\overrightarrow{x})$ is an arbitrary SQL query over the sources and $\Psi(f(\overrightarrow{x}), \overrightarrow{x})$ is a conjunctive query over the target ontology. This mapping form proposes a solution to the impedance mismatch problem through the use of skolem functions, i.e. f in the previous definition. In the following mapping M_2, two skolem functions are introduced, i.e. d and l, to denote instances of concepts, respectively Lab and Drug.

Example 3: Consider the mappings for Example 1 and 2:

M_1: SELECT labId, name FROM lab	$\rightsquigarrow$	Lab(l(labId)) labName(l(labId), labName)
M_2: SELECT d.cip, d.name, d.lab FROM drugs d,drugTherapeutics dt,therapeutics t WHERE dt.classId=t.classId AND d.cip= dt.cip AND className like 'Antitussive'	$\rightsquigarrow$	Antitussive(d(cip)) drugName(d(cip), name) producedBy(d(cip),l(labId))

M_2 characterizes our integration approach since it introduces a new concept, *Antitussive*, which is not present in the original ontology of Example 2. After M_2's validation, the *Antitussive* concept will be integrated in the ontology and therefore will be considered just like any other concepts of this ontology.

3 Mapping Generation

Intuitively, our method takes a limited set of user generated mappings as an input and derives new mappings based on uncovered constants of DB instances. The effectiveness of this mapping generation depends on the semantic information associated to the constant values. We consider that in DBs containing classifications or terminologies, e.g. medicine, geography or ecology, this semantic information is generally available and valuable.

Our approach requires to distinguish between two kinds of mappings denoted M_d and M_u. Using the notation M_u, we denote the mappings that have been defined by **end-users** or created from a computer system. We denote by M_d the set of mappings **derived** by our solution. We state that $M = M_d \cup M_u$ and $M_d \cap M_u = \emptyset$. For a mapping M, we denote with *LHS(M)*, left hand-side of a mapping M, i.e. the (SQL) query over the sources, and denote by *RHS(M)*, right hand-side of M, i.e. the Conjunctive Query (CQ) over elements of the ontology. In this paper, we restrict *LHS(M)* to SPJ (Select Project Join) queries with equalities. For instance, we do not consider aggregation functions or GROUP BY in *LHS(M)*. Our system exploits metadata of the DBs in order to derive new mappings. This information corresponds to the most commonly used constraints over the relational model: primary and foreign key constraints.

Another aspect that needs to be considered is equivalence of CQ which plays an important role in discovering constant values with high semantic information. For instance, considering the LHS(M_2) SQL query, we can create the equivalent following query, denoted LHS(M_2'):

SELECT d.cip, d.name, d.lab FROM drugs d, drugTherapeutics dt WHERE d.cip=dt.cip AND dt.classId=1.

The queries LHS(M_2) and LHS(M_2') are equivalent since their answer sets are identical for a given instance of the drug DB. Both SQL queries present a constant, i.e. 'Antitussive' in M_2 and '1' in M_2', which do not have the same 'direct' semantic information. But in the context of the query and DB schema, we can associate an equivalent semantic information to '1' and 'Antitussive' since both columns are related by a primary key constraint. Similar equivalence relationships can be discovered using foreign key constraints.

We now present a structure Ω which stores the information associated to the occurence of constant values in the conjunctive query of a mapping. This structure Ω corresponds to an ordered set of triples $\{\omega_1, .., \omega_n\}$ where ω_i is defined as a triple $\langle R_j, A_k, c\rangle$. In this triple, R_j corresponds to a relation of the source, A_k is an attribute of R_j with $1 \leqslant k \leqslant |R_j|$ (arity of the relation R_j), and c is the constant value associated to A_k in the conjunctive query and thus belongs to Δ. In this paper, we are interested in computing such structures for both M_u and M_d and thus denote them respectively with Ω_u and Ω_d. Once we have generated the Ω_u from a given *LHS(*M_u*)*, we start a second step aiming to derive new mappings. The method searches for new constant values for Ω_d which are different from the ones used in Ω_u. In order to find these values, we execute an aggregation query based on *LHS(*M_u*)*. This query returns the number of instances retrieved for a rewriting of *LHS(*M_u*)* for each constant value discovered and sorts the results in descending order on the number of instances of these groups. This ordering is justified by the assumption that concepts or roles with the most instance assertions should be the most pertinent in the context of creating an ontology. The aggregation for *LHS(*M_u*)* of M_2 and M_2' correspond respectively to the following SQL queries:

(1) SELECT className, count(*) FROM therapeutics GROUP BY className HAVING className not like 'Antitussive' ORDER BY count(*) DESC;
(2) SELECT classId, count(*) FROM drugTherapeutics GROUP BY classId HAVING classId <> 1 ORDER BY count(*) DESC;

Queries with $|\Omega_u| \geqslant 2$, i.e. the size of the set Ω_u, may retrieve very large sets of candidate values, i.e. based on the cartesian product of all constant values of Δ for attributes in Ω_u. In order to minimize the number of irrelevant mappings generated, our system interacts with the end-user to select relevant attributes. Basically, for a given mapping, it enables an end-user to select a subset of relations and attributes of Ω_u for which new constants should be searched.

The interaction with the end-user is handled by a GUI taking the form of a list of relation, $\Omega_u.R_j$, and attribute, $\Omega_u.A_k$ couples. Each entry of the list is associated to a check box component. This enables to select/deselect attributes in an effective way. The GUI also interacts with the end-user's selection to display the number of mappings that will be automatically generated if this selection is validated. Finally, when $|\Omega_u| = 0$, the system does not search for new mappings as the *LHS(*M_u*)* already retrieves all instances of a given relation R_i of S.

Example 4: In our medical example, the *LHS(*M_u*)* of a mapping may contain two constants: therapeutic class name and laboratory name, i.e. 'className' (resp. 'labName') attribute in the 'therapeutics' (resp. 'labs') relation. Because $|\Omega_u| \geqslant 2$ for this mapping, the system asks the end-user which attributes need to be searched. For instance, the end-user may decide to keep the same laboratory name and retrieve new therapeutic classes (or inverse) or retrieve combinations for all laboratories and therapeutic classes (creating a large set of M_d mappings).

The results obtained from the execution of queries (1) and (2), respectively enable to generate the following *LHS(*M_d*)*:

(3) SELECT d.cip, d.name, d.lab FROM drugs d, drugTherapeutics dt, therapeutics t WHERE d.cip=dt.cip AND dt.classId=t.classId AND className like 'Analgesic';
(4) SELECT d.cip, d.name, d.lab FROM drugs d, drugTherapeutics dt WHERE d.cip=dt.cip AND dt.classId=2.

Both queries are being generated using a rewriting of *LHS(M_u)* by substituting constant values of Ω_u with the new constant values of Ω_d. We now consider these M_d queries as candidate queries for ontology mappings.

4 Providing Labels to Concepts and Roles

In the previous section, we presented a solution to generate *LHS(M_d)*, i.e. source queries, based on predefined mappings. In cases a M_u mapping introduces new ontology elements, e.g. 'Antitussive' concept in M_2, then it is necessary to search for relevant ontology element labels for M_d. For instance what *RHS(M_d)* can we associate to the SQL queries (3) or (4)?

In order to perform this task, the system searches for relationships between the constants in *LHS(M_u)* and the ontology elements introduced in *RHS(M_u)*. Then it applies these relationships on the *LHS(M_d)* queries to discover ontology element labels to complete the *RHS(M_d)* queries. We distinguish two main relationships between constants: (i) lexicographic equivalence, i.e. a constant and an ontology element correspond to the same string. We also consider some forms of simple lexicographic transformations and concatenation over Ω_u constant values. (ii) non lexicographic equivalence, i.e. a constant value from the SQL query does not correspond to any label of the ontology elements. In this situation, we are searching for other forms of equivalence, e.g. synonymy or hyponymy.

When an equivalence relationship is detected, then the symbol for the ontology element is obvious and can be computed from the constant value. In our running example, a lexicographic equivalence is detected in mapping M_2. Thus we are able to easily propose a label for the concept of the derived mappings associated to the LHS queries (3) and (4): *Analgesic*. Moreover, we are now able to complete the mapping M_3 from M_2 by substituting *Antitussive* with *Analgesic* in the RHS conjunctive query of M_2.

M_3 : SELECT d.cip, d.name, d.lab
FROM drugs d, drugTherapeutics dt
,therapeutics t WHERE d.cip=dt.cip
AND dt.classId=t.classId
AND className like 'Analgesic'

$\rightsquigarrow$ Analgesic(d(cip))
drugName(d(cip), name)
producedBy(d(cip),l(labId))

We adopt the same approach if multiple constants appear in M_u (i.e. $|\Omega_u| \geqslant 2$). That is if one of the constant values in Ω_u equals one of *LHS(M_u)*, then we mark its triple (R_k, A_n, c) and for all derived *LHS(M_d)*, we set the ontology element symbol with the constant value of the marked Ω_d.

When no lexicographic equivalence holds, it is necessary to discover a relationship between some or all of the constant values and the ontology element

labels in the end-user mapping. Once these relations have been discovered, we can apply them to the corresponding constants of M_d and find labels for the associated ontology elements. In order to perform this discovery, we exploit information coming from some background knowledge, e.g. WordNet (an electronic lexical database). Using WordNet, we consider only nouns as label candidates and we use the Java WordNet Library (http://jwordnet.sourceforge.net) to recognize variants of the same noun obtained by applying the usual inflection rules. Our method considers WordNet as a graph where vertices correspond to nouns or synsets. The edges between these nodes are relations of the kind hyponyms, hypernyms, meronyms, etc.. Using this graph representation, it is possible to navigate through this background knowledge and find relations between nodes. The labeling solution is decomposed into two algorithms: `getRelation` and `getLabel` which are detailed below.

In a nutshell, the `getRelation` algorithm accesses a list of nouns that match the labels of the newly introduced elements of $RHS(M_u)$. For instance 11 synsets are available for the noun 'Man'. Then the algorithm uses all the information available from Ω_u, i.e. relation, attribute names and constants, to find the most appropriate synset. The selection of the most appropriate one is performed using a score function. Once a synset has been selected, we need to search for the most appropriate relations, i.e. hypernym, hyponym, meronym,etc., available. Again, this is done using a function that scores for each relation the number of matches with the terms of Ω_u. Finally the most relevant relation is returned. Several heuristics are added to our `getRelation` algorithm. For instance, if no hypernyms, hyponyms or meronyms, etc. are found using WordNet, then we return a '`null`' relation which implies that constant values of Ω_u are proposed as ontology labels. Also, if the scores of all or some the hypernyms, hyponyms, meronyms, etc. relations are equal then we return the hyponym relation.

Once we have characterized the relation between the ontology symbol and the query of M_u, we use this relation with the constant values of M_d to find a set of symbol candidates for M_d. This task is performed by the `getLabel` algorithm for each end-user selected Ω_u element. The inputs of this algorithm are the Ω_d elements and the WordNet relation returned from `getRelation`. A first step is to retrieve a set of words corresponding to the constant value of the Ω_d element. Then the most adequate noun is selected and a set of synsets is retrieved from WordNet based on the selected noun and the previously discovered WordNet relation. Then synsets are rated in a way similar to the `getRelation` algorithm, i.e. using a score function that searches for matches between synset descriptions,labels and elements of Ω_d triples. Finally a set of synset labels is returned by the `getLabel` function. In cases where several constant values have been selected by the end-user, several execution, one for each selected Ω_d triple selected, is performed and the set of candidate labels then corresponds to the union of the returned labels.

Once a derived mapping M_d is completed, i.e. both the LHS and RHS queries have been generated, the next logical operation is to make it persitent in the integration system. Basically, this is performed by storing the derived mapping in a mapping repository and recording the new ontology elements in the TBox. But

before performing these operations, we need to make sure that the derived mappings satisfy the end-user's intention. The consideration of this aspect prevents the system from generating large TBoxes where only a subset of the concepts and roles are relevant to the ontology designer. This is the case if one wants to design an ontology that only considers a restricted part of the domain of an DB, e.g. design an over-the-counter drug ontology from a complete drug database. Hence it is necessary to consider only a relevant subset of the mappings that can be effectively derived. Basically, this is usually a consequence of an under restricting of M_u's SQL query. In order to correct this situation, the only reliable source of information is the end-user.

We propose a user-friendly graphical user interface (GUI) for the acceptance of mappings which enables an effective scan and selection of candidate mappings as well as an easy solution to refine SQL queries associated to these mappings. This GUI displays: (1) a pattern of LHS SQL query of the mappings where all constant values are substituted by a symbol (Cx). (2) an evaluation of the number of mapping to process and number of emerging concepts and roles. (3) a simplified view of mappings via pairs consisting of a set of constant values and a set of ontology element labels. (4) a check box for each mapping view which enables to select a mapping. (5) text areas where the end-user can enrich the FROM and WHERE clauses of the aggregation SQL query responsible for finding constant values.

Based on this GUI, it is clear that the system proposes two different approaches to refine, adopt or reject a set of mappings. A first approach consists in selecting, via the check box, a set of satisfying mappings. This approach is useful when the number of proposed mappings is low, in practice we have studied that the upper bound is about thirty mappings. We have seen in the previous sections that our solution can easily derive hundreds or thousands mappings from certain classifications. In this situation, it is not manageable for the end-user to manually check such an amount of check boxes. Hence, we propose a text area enabling to restrict the set of discovered constant values. This is performed by introducing new tables in the FROM clause and conditions in the WHERE clause of the aggregation SQL query.

5 Related Work

The interest of discovering mappings between relational databases and ontologies relates to an interdisciplinary research topic including both database and Semantic Web communities. The infatuation surrounding the Semantic Web motivates several research teams to study cooperations between the domains of databases and knowledge bases represented in a DL. As stated in [6], although the expressivity of DLs underlying OWL and of relational dependencies is clearly different, the schema languages of the two are quite closely related.

MASTRO-I [3] can be considered as the reference implementation for the OBDA project. The main difference with our system is that MASTRO-I does not enable to introduce new concepts and roles via mappings and that no internal solution enables to derive mappings automatically or semi-automatically.

Anyhow, the domain of deriving mappings between relational databases and ontologies has been the subject of several papers. In Maponto tool [1], the authors propose a solution that enables to define complex mappings from simple correspondences. Like our system, this approach expects end-users or an external software to provide mappings and then generate new ones. The main difference between the two systems is that Maponto is being provided with a set of relational databases and an existing ontology, while in our solution the ontology is defined incrementally via validation of the mappings. Thus Maponto does not require a label generation approach. Another system, MARSON [4], discovers simple mappings by classifying the relations of a database schema and validate the mapping consistency that have been generated. Like Maponto, these systems require that the target ontology is provided.

6 Conclusion

In this paper, we have introduced an OBDA system extended with an ontology integration approach, i.e. concepts and roles can be integrated in the ontology via execution of mappings. We have also proposed a mapping generation solution consisting of three distinct operations: (i) generation of mappings from a previously defined mapping containing at least one constant, (ii) providing labels to ontology elements associated to derived mappings and (iii) an interactive solution to select/refine a set of candidate mappings.

We are currently working on an extension of the labeling solution to handle domain specific data, e.g. scientific data. In order to tackle this issue, we are aiming to complement our background knowledge provider with access to open data sets availbale on the Semantic Web, e.g. Open Data Movement.

References

1. An, Y., Borgida, A., Mylopoulos, J.: Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3761, pp. 1152–1169. Springer, Heidelberg (2005)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): Description Logic Handbook. Cambridge University Press, Cambridge (2003)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Mastro-i: Efficient integration of relational data through dl ontologies. In: Proc. of the 2007 Description Logic Workshop, vol. 250, pp. 227–234 (2007)
4. Hu, W., Qu, Y.: Discovering simple mappings between relational database schemas and ontologies. In: ISWC, pp. 225–238 (2007)
5. Jablonski, S., Curé, O., Rehman, M.A., Volz, B.: Dalton: An infrastructure for scientific data management. In: ICCS (3), pp. 520–529 (2008)
6. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between owl and relational databases. In: WWW, pp. 807–816 (2007)
7. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)