A Semantic Mapping System for Bridging the Gap between Relational Database and Semantic Web

Linhua Zhou
School of Computer Science,
Zhejiang University, CN
zlhyjy@zju.edu.cn

Huajun Chen
School of Computer Science,
Zhejiang University, CN
huajunsir@zju.edu.cn

Yu Zhang, Chunying Zhou
School of Computer Science,
Zhejiang University, CN
{yzh,zcy1982}@zju.edu.cn

Abstract

Recently, there has been a growing need for integrating legacy relational databases with semantic web ontologies, however, experience in building such applications has revealed a gap between semantic web languages and relational data model. We present a formal mapping system to bridge the gap and study the problem of reasoning and query answering using view underlying the mapping system. Particularly, we consider a special mapping called "constraint mapping" between database integrity constraints and OWL axioms. We then propose an approach to incorporating these OWL axioms and description logic reasoning into query rewriting using views to enable the algorithm to answer more types of queries. The approach has been used to develop a pilot data integration application for neuroscience community.

Motivation

Semantic Web provides enhanced capabilities for data integration by making data semantics explicit through machine-understandable ontologies. It is based upon Resource Description Framework (RDF) and Web Ontology Language(OWL), the semantic web languages proposed by W3C Consortium. Relational databases are commonly used to build the backbone of the modern information systems. With the popularity of semantic web, there is an ongoing increase of interest and endeavor in transforming and integrating legacy relational databases into semantic web. For example, in life science community, researchers are trying to transform legacy neuroscience relational databases into RDF/OWL format to enable integrative query answering and reasoning(Hugo Y.K. 2007). However, experience in building such applications has revealed a gap between semantic web languages and relational data model; this gap becomes more noticeable if we consider expressive languages such as OWL-DL or OWL-Full.

To understand the problems in specific, consider the following example. For a bookstore application, one might require each book to have at least one book author. In a relational database, this could be captured by a foreign key constraint stating that, for each tuple in book table, an associated tuple in author table must exist. Such a constraint would be used to ensure data integrity during database updates. In an OWL knowledge base, the same requirement could be captured by an *owl:someValuesFrom* axiom asserting that for all book, they have at least one author. However, the key difference is that the OWL axiom would not be used to check data integrity. One can add a new book into the knowledge base without specifying an author for that book and the knowledge base system would not consider it as an error, rather, that book would be simply inferred to have some unknown author. As a matter of fact, OWL knowledge base consider the axiom as a kind of TBox knowledge for reasoning.

More issues arise while mapping legacy relational databases to OWL ontologies. Most available approaches merely consider the mappings between tables/columns and classes/propertes, with no consideration of database integrity constraints which are actually important information and knowledge. Problem comes up while one tries to do query answering and reasoning. For the same example, if new books with no authors specified are added into the OWL knowledge base (OWL-KB) and one issues a query of all books having at least one author, the OWL-KB would discard all of those newly added books. As can be seen in our approach, although the constraints, which are mapped to different OWL axioms, are not used by OWL-KB to check data integrity, they can absolutely be used to enhance query answering and rewriting.

This paper presents a view-based mapping mechanism to bridge the gap between OWL ontologies and schemata of relational data model in a seamless manner, and study the problem of *answering queries using views* underlying this mapping system. In particular, we consider a special mapping called "constraint mapping" between database integrity constraints and OWL axioms. We then propose an approach to incorporating these OWL axioms and description logic reasoning into query rewriting using views to enable the algorithm to answer more types of queries. The approach has been used to develop a pilot data integration application for neuroscience community. For formal discussion, we focus attention on the $\mathcal{SHIQ}$(Ian Horrocks 2003) description logic language, which is the underlying formalism for semantic web ontology languages.

Preliminaries

$\mathcal{SHIQ}$ Description Logic

Syntax. Given $\mathcal{R}$ as a finite set of transitive and inclusion axioms with normal role names N_R, a $\mathcal{SHIQ}$*-role* is

Table 1: Semantics of $\mathcal{SHIQ} - \mathcal{KB}$

Interpretation of Concepts
$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, C) \neq \emptyset\}$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, \neg C) = \emptyset\}$
$(\leqslant nS.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp R^{\mathcal{I}}(x, C) \leqslant n\}$
$(\geqslant nS.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp R^{\mathcal{I}}(x, C) \geqslant n\}$
Interpretation of Axioms
$(C \sqsubseteq D)^{\mathcal{I}} = C^{\mathcal{I}} \subseteq D^{\mathcal{I}}, (R \sqsubseteq S)^{\mathcal{I}} = R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$(\mathsf{Trans}(R))^{\mathcal{I}}$ $= \{\forall x, y, z \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, y) \cap R^{\mathcal{I}}(y, z) \rightarrow R^{\mathcal{I}}(x, z)\}$
$\sharp R$ is the number restriction of a set R and $R^{\mathcal{I}}(x, C)$ is defined as $\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} and y \in C^{\mathcal{I}}\}$.

either some $R \in N_R$ or an *inverse role* R^- for $R \in N_R$. $\mathsf{Trans}(R)$ and $R \sqsubseteq S$ represent the transitive and inclusion axioms, respectively, where R and S are roles. A simple role is a $\mathcal{SHIQ}$*-role* that neither its sub-roles nor itself is transitive. Let N_C be a set of *concept names* , the set of $\mathcal{SHIQ}$-concepts is the minimal set such that every concept $C \in \mathcal{N}_C$ is a $\mathcal{SHIQ}$-concept and for C and D are $\mathcal{SHIQ}$-concepts, R is a role, S a simple role and n a positive integer, then $(\neg C)$, $(C \sqcap D)$, $(C \sqcup D)$, $(\exists R.C)$, $(\forall R.C)$, $(\leqslant nS.C)$ and $(\geqslant nS.C)$ are also $\mathcal{SHIQ}$-concepts. Therefore we have a knowledge base $\mathcal{KB}$ is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ where (1) a TBox $\mathcal{T}$ is a finite set of axioms representing the concept inclusions with the form $C \sqsubseteq D$; (2) an ABox $\mathcal{A}$ is a finite set of axioms with individual names N_I and the form $C(x)$, $R(x, y)$ that consists equality-relations x is (un)equal y.

Semantics. The semantics of $\mathcal{KB}$ is given by the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of $\mathcal{I}$) and the function $\cdot^{\mathcal{I}}$ in the Table (Ian Horrocks 2003). It is well known that $\mathcal{KB}$ is satisfiable with respect to interpretation $\mathcal{I}$ *iff* $\mathcal{KB}^{\mathcal{I}}$ is satisfiable in first order logic. The satisfiability checking of $\mathcal{KB}$ in expressive Description Logic (DL) is performed by reducing the subsumption, and the reasoning over TBox and role hierarchy can be reduced to reasoning over only role hierarchy. The interpretation $\mathcal{I}$ is the model of $\mathcal{R}$ and $\mathcal{T}$ if for each $R \sqsubseteq S \in \mathcal{R}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ and for each $C \sqsubseteq D \in \mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Answering Queries Using Views

A data integration system usually consists of three components: source schemata, target global schema, and mappings between them. There are two main approaches for specifying the mapping: in the local-as-view (LaV) approach (Lenzerini 2002) the source structures are defined as views over the global schema; on the contrary in the global-as-view (GaV) approach each global concept is defined in terms of a view over the source schemata. A more general approach is called global-local-as-view (GLaV) which combines the expressivity of both LaV and GaV. Answering queries using views (Halevy 2001) aims to find efficient means for answering queries posed by the global schema on the basis of the mapping specification and source data. Query rewriting using views aims to compute a rewriting

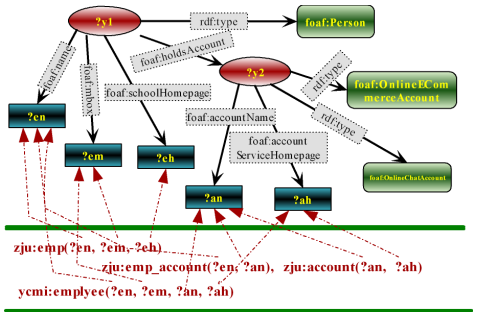

Figure 1: $\mathcal{SHIQ}$-$\mathcal{RDM}$ Mapping Scenario.

of the query in terms of the views and then evaluates the rewriting directly against the source data. The queries in actual applications are usually conjunctive queries. A view is a named query. In this paper, we consider conjunctive query for $\mathcal{SHIQ}$.

Definition 1 ($\mathcal{SHIQ}$ **Conjunctive Queries)** *Let N_V be a countably infinite set of variables disjoint from N_C, N_R, and N_I. An atom is an expression $A(v)$ (concept atom) or $R(v, v')$ (role atom), where A is a concept name, R is a role, and $v, v' \in N_V$. A conjunctive query q is a non-empty set of atoms in the form:*

$q(\bar{X}) : e_1(\bar{X}_1, \bar{Y}_1), \ldots, e_n(\bar{X}_n, \bar{Y}_n)$*, where*

- *q belongs to a new alphabet $\mathcal{Q}$ of queries that is disjoint from N_C, N_R, N_I and N_V.*
- *$e_1(\bar{X}_1, \bar{Y}_1), \ldots, e_n(\bar{X}_n, , \bar{Y}_n)$ are called the subgoals in the query body, and are either a concept atom in the form of $A(v)$ or a role atom in the form of $R(v, v')$.*
- *$\bar{X}_1, \ldots, \bar{X}_n$ are either variables from N_V or constants from N_I, and $\bar{X} \subseteq \bar{X}_1 \cup \ldots \cup \bar{X}_n$ are called distinguished variables. $\bar{Y}_1, \ldots, \bar{Y}_n$ are called existential variables.*

As like in conventional literatures(Halevy 2001), the *distinguished variables* must have a value binding while doing query evaluation, instead, *existential variables* do not have to bind to any instance or data values.

$\mathcal{SHIQ}$-$\mathcal{RDM}$ Mapping System

$\mathcal{SHIQ}$-$\mathcal{RDM}$ View

The basic mapping system consists of a mapping formalism between a target $\mathcal{SHIQ}$ ontology and a set of source relational schemata that employs the form of LaV (i.e., each relational predicate is defined as a view over the ontology).

Definition 2 ($\mathcal{SHIQ}$-$\mathcal{RDM}$ **View)** *A $\mathcal{SHIQ}$-$\mathcal{RDM}$ view is in this form: $\mathfrak{R}(\bar{\mathbf{X}}) : -\mathfrak{B}(\bar{\mathbf{X}}', \bar{\mathbf{Y}}')$ where:*

- *$\mathfrak{R}(\bar{\mathbf{X}})$is called the head of the view, and $\mathfrak{R}$ is a relational predicate.*
- *$\mathfrak{B}(\bar{\mathbf{X}}', \bar{\mathbf{Y}}')$ is called the body of the view, and is a $\mathcal{SHIQ}$ conjunctive query over the ontology.*
- *The $\bar{\mathbf{X}} \subseteq \mathbf{X}'$ is the set of distinguished variables, $\bar{\mathbf{Y}}'$ is the set of existential variables.*

Figure 1 is a $\mathcal{SHIQ}$-$\mathcal{RDM}$ mapping scenario consisting of the target FOAF ontology[1] and two source relational data

[1]The FOAF project: `http://www.foaf-project.org/`.

Table 2: Constraint Mapping. T_1, T_2 are relational tables, and are mapped to class A, C respectively.

Primary Key Constraint : $T_1 \Rightarrow$ Axiom: $A \subseteqq \exists hasPK.\top \sqcap \leq 1 hasPK.\top$ Formal Semantics: $\forall x : [A(x) \mapsto \exists y : hasPK(x,y)]$ AND $\forall x, y1, y2 : [A(x) \wedge hasPK(y1) \wedge hasPK(y2) \mapsto y1 \approx y2]$
Foreign Key Constraint: T_1,$T_2 \Rightarrow$ Axiom: $A \subseteqq \exists R.C$ Formal Semantics: $\forall x : [A(x) \mapsto \exists y : R(x,y) \wedge C(y)]$
$\leqslant n$-Cardinality Constraint: T_1,$T_2 \Rightarrow$ Axiom: $A \subseteqq \leqslant nR.C$ Formal Semantics: $\forall x : [A(x) \mapsto \sharp R^{\mathcal{I}}(x,C) \leqslant n\}]$
$\geqslant n$-Cardinality Constraint: T_1,$T_2 \Rightarrow$ Axiom: $A \subseteqq \geqslant nR.C$ Formal Semantics: $\forall x : [A(x) \mapsto \sharp R^{\mathcal{I}}(x,C) \geqslant n\}]$
n-Cardinality Constraint: T_1,$T_2 \Rightarrow$ Axiom: $A \subseteqq \leqslant nR.C \sqcap \geqslant nR.C$ Formal Semantics: $\forall x : [A(x) \mapsto \sharp R^{\mathcal{I}}(x,C) = n\}]$

sources "zju" and "$ycmi$". Example 1 illustrates the corresponding $\mathcal{SHIQ}$-$\mathcal{RDM}$ views. Note we use RDF triple syntax to describe concept atom and role atom.

Example 1 ($\mathcal{SHIQ}$-$\mathcal{RDM}$ View Examples)

```
ZJU Source
v1:  zju:emp(?en,?em,?eh) :--
          (?y1,rdf:type,foaf:Person),
          (?y1,foaf:name,?en),
          (?y1,foaf:mbox,?em),
          (?y1,foaf:schoolHomepage,?eh).
v2:  zju:emp_account(?en,?an) :--
          (?y1,rdf:type,foaf:Person),
          (?y1,foaf:name,?en),
          (?y1,foaf:holdsAccount, ?y2),
          (?y2, rdf:type, foaf:OnlineChatAccount),
          (?y2, foaf:accountName,?an).
v3:  zju:account(?an,?ah) :--
          (?y1, rdf:type, foaf:OnlineChatAccount),
          (?y1, foaf:accountName, ?an),
          (?y1, foaf:accountServiceHomepage, ?ah).

YCMI Source:
v4:  ycmi:emp(?en,?em,?an,?ah) :--
          (?y1, rdf:type, foaf:Person),
          (?y1, foaf:name, ?en),
          (?y1, foaf:mbox, ?em),
          (?y1, foaf:holdsAccount, ?y2),
          (?y2, rdf:type, foaf:OnlineEcommerceAccount),
          (?y2, foaf:accountName, ?an),
          (?y2, foaf:accountServiceHomepage, ?ah)
```

Constraint Mapping

Database integrity constraints such as *Primary Key, Foreign Key, and Cardinality Constraint* are actually important information. However, most of available mapping approaches merely consider the mappings between tables/columns and classes/propertes disregarding the knowledge captured in constraints. We propose a special mapping called "constraint mapping" between database integrity constraints and $\mathcal{SHIQ}$ axioms. We then show in the next section an approach to incorporating these $\mathcal{SHIQ}$ axioms and description logic reasoning into query rewriting using views to enable the algorithm to answer more types of queries. Table 2 illustrates the set of constraint mapping rules for generating $\mathcal{SHIQ}$ axioms.

As examples, suppose ?en is the primary key of table *zju:emp(?en,?em,?eh)*, *zju:emp(?en,?an)* has a foreign key relationship with *zju:account(?an,?ah)* on ?an, and the database requires that all person has exactly 2 accounts. We could generate the following $\mathcal{SHIQ}$ axioms.

Example 2 (Generated $\mathcal{SHIQ}$ Axioms)

1. *foaf:Person* $\subseteq \exists$*foaf:name* $\sqcap \leq 1$*foaf:name*
2. *foaf:Person* $\subseteq \exists$*foaf:holdsAccount.foaf:Account*
3. *foaf:Person* $\subseteq \leq 2$*foaf:holdsAccount.foaf:Account*
4. *foaf:Person* $\subseteq \geq 2$*foaf:holdsAccount.foaf:Account*

Rewriting $\mathcal{SHIQ}$ Queries using $\mathcal{SHIQ}$-$\mathcal{RDM}$ Views

In this section, we present an algorithm to rewrite conjunctive $\mathcal{SHIQ}$ queries into SQL queries on the basis of the mapping system presented in previous section.

The Problem

Given a $\mathcal{SHIQ}$ TBox $\mathcal{T}$, a set of source relational instances $\mathcal{I}$, a set of $\mathcal{SHIQ}$-$\mathcal{RDM}$ views, plus a set of constraint mappings $\mathcal{CM}$, the central problem we are interested in is *how to compute answers to $\mathcal{SHIQ}$ queries by rewriting the ontology queries into relational queries, on the basis of the mapping system.*

Example 3 ($\mathcal{SHIQ}$ Conjunctive Query Example) *Note again, we use RDF triple syntax to describe the concept atoms and role atoms in the query body.*

```
Q1: q(?en,?em,?an,?ah):- (?y1 rdf:type foaf:person),
       (?y1 foaf:name ?en), (?y1 foaf:mbox ?em),
       (?y1 foaf:holdsAccount ?y2),
       (?y2 rdf:type foaf:OnlineAccount),
       (?y2 foaf:accountName ?an),(?y2 foaf:homepage ?ah).
```

Hereinbelow, we present a query rewriting approach which can be generally divided into three stages: 1)Reasoning over views; 2) Splitting views into smaller mapping rules; 3)Rewriting queries using mapping rules.

Reasoning over Views

In our approach, we try to enable description logic reasoning in the query rewriting process. This goal is achieved by including an extra reasoning process over the $\mathcal{SHIQ}$-$\mathcal{RDM}$ views. In this process, a set of rules based on $\mathcal{SHIQ}$ TBox axioms, including the axioms generated by constraint mappings, are applied on the original view definitions. First of all, we formally specify these rules as follows.

Definition 3 (View Reasoning Rules) *Let $x, x1, x2, y$ denote either variable names from N_V or constant names from N_I, let $A(x)$ and $R(x1, x2)$ denote concept atom and role atom, and let $\mathcal{B}$ denote the set of atoms in the view body.*

1. *$\subseteqq_C$-rule: **IF** a)$A \subseteqq A'$, b) $A(x) \in \mathcal{B}$ c)$A'(x) \notin \mathcal{B}$ **THEN** ADD $A'(x)$ into $\mathcal{B}$.*
2. *$\subseteqq_R$-rule: **IF** a)$R \subseteqq R'$, b) $R(x1, x2) \in \mathcal{B}$, c)$R'(x1, x2) \notin \mathcal{B}$ **THEN** ADD $R'(x1, x2)$ into $\mathcal{B}$.*
3. *$\exists$-rule: **IF** a)$A \subseteqq \exists S.C$, b)$A(x) \in (B)$, c)There is no y such that $S(x, y) \in \mathcal{B}$ and $C(y) \in \mathcal{B}$ **THEN** ADD both $S(x, y)$ and $C(y)$ to (B) where y is an new existential variable.*

4. *$\forall$-rule:* ***IF*** *a)$A \subseteq \forall S.C$, b)$A(x1) \in \mathcal{B}$, c)There is a $x2$ such that $S(x1, x2) \in \mathcal{B}$ and $C(x2) \notin \mathcal{B}$* ***THEN*** *ADD $C(x2)$ to $\mathcal{B}$.*
5. *$\geq_n$-rule:* ***IF*** *a)$A \subseteq \geq nS.C$, b)$A(x) \in \mathcal{B}$, c) x has no n S-related $y1, ..., yn$ such that $C(yi) \in \mathcal{B}$ and $y_i \neq y_j$ for $1 \leq i \leq j \leq n$* ***THEN*** *ADD into $\mathcal{B}$ with n new role atom $S(x, yi)$ and n new concept atom $C(yi)$ for all $1 \leq i \leq n$.*
6. *$\leq_n$-rule:* ***IF*** *a)$A \subseteq \leq nS.C$, b)$A(x) \in \mathcal{B}$, c) x has more than n S-related y such that $C(y) \in \mathcal{B}$,d)There are two S-related $y1, y2$ such that $C(y1) \in \mathcal{B}$ and $C(y2) \in \mathcal{B}$ with $y1 \neq y2$* ***THEN*** *Replace all concept atoms $A'(y2) \in \mathcal{B}$ with $A'(y1)$ and all role atoms $R'(x, y2) \in \mathcal{B}$ with $R'(x, y1)$.*
7. *Inv-rule:* ***IF*** *a) R^- is the inverse role of R, b)$R(x1, x2) \in \mathcal{B}$,* ***THEN*** *ADD $R^-(x2, x1)$ into $\mathcal{B}$.*

For example, suppose the TBox has the following axioms, applying these axioms together with the axioms generated by constraint mappings (c.f. Example 2) to v4 in Example 1 results in the extended view as below.

Example 4 (TBox Axioms)

1. *foaf:OnlineChatAccount $\subseteq$ foaf:Account*
2. *foaf:accountServiceHomepage $\subseteq$ foaf:homepage*

Example 5 (Extended View Example)

```
v4:  ycmi:emp(?en,?em,?an,?ah) :--
          (?y1, rdf:type, foaf:Person),
          (?y1, foaf:name, ?en),(?y1, foaf:mbox, ?em),
          (?y1, foaf:holdsAccount, ?y2),
          (?y2, rdf:type, foaf:OnlineEcommerceAccount),
          (?y2, foaf:accountName, ?an),
          (?y2, foaf:accountServiceHomepage, ?ah),

          (?y2, rdf:type, foaf:Account),
          (?y2, foaf:homepage, ?ah),
          (?y1, foaf:holdsAccount, ?y3),
          (?y3, rdf:type, foaf:Account).
```

This extra reasoning process is necessary and essential because it directly incorporates the constraints information and TBox knowledge into the view definitions, and enables the rewriting algorithm to answer more types of query. For example, obviously Q_1 can not be answered by using the original views because the query makes a reference to foaf:OnlineAccount and foaf:homepage which do not appear in any original view definitions at all. For another example, by incorporating axiom 3,4 in Example 2 into the view, the system can answer the query of all person who has exact two accounts. Most importantly, because all TBox knowledge and constraints information has been encoded into the views as new *concept atoms or role atoms*, it greatly facilitates the implementation and improves the performance of the query rewriting algorithm since the algorithm does not need to consider the TBox knowledge and the constraints information anymore while doing query translation.

Splitting Views

In the second step, the extended views are splitted into smaller rules called *Class Mapping Rules*. The purpose is to make it easier to match and replace the bodies of queries and views in the rewriting process.

Definition 4 (Class Mapping Rule $\mathcal{CMR}$) *A $\mathcal{CMR}$ rule is in the form*

$t_1(\bar{X_1}), ..., t_n(\bar{X_n}) :- A(x), R_1(x, y_1), ..., R_m(x, y_m)$ *where*

- $t_1(\bar{X_1}), .., t_n(\bar{X_n})$ *are relational predicates.*
- *A(x) is a concept atom, $R_1(x, y_1), ..., R_m(x, y_m)$ are role atoms.*

Intuitively, a $\mathcal{CMR}$ rule defines a mapping from the relational predicate to a subset triples (triple group) of view body.

Definition 5 (Triple Group) *A triple group of a view body is a set of triples that have the same subject. For example, the first four triples of v4 can be considered as a triple group, the next three triples can be viewed as another triple group. In Example 6, R1 defines the mapping from ycmi:emp(?en,?em,?an) to the first triple group, and R2 defines the mapping to the second triple group.*

Triple group serves similar purpose as the MCD proposed in the MiniCon algorithm (Rachel Pottinger 2001). A triple group represents a fragment of the view body which can later be matched and combined more easily.

Example 6 (Class Mapping Rule Examples)

```
R1: ycmi:emp(?en,?em,?an):-
        (y1,rdf:type,foaf:Person),
        (y1,foaf:name,?en), (y1,foaf:mbox,?em).
R2: ycmi:emp(?an,?ah):-
        (y2,rdf:type,foaf:OnlineEcommerceAccount),
        (y2,rdf:type,foaf:OnlineAccount),
        (y2,foaf:accountName,?an),(y2,foaf:homepage,?ah).
        (y2,foaf:accountServiceHomepage,?ah),
R3: zju:emp(?en,?em,?eh):-
        (y1,rdf:type,foaf:Person),
        (y1,foaf:name,?en), (y1,foaf:mbox,?em),
        (y1,foaf:schoolHomepage,?eh),(y1,foaf:homepage,?eh).
R4: zju:emp_account(?en,?an):-
        (y1,rdf:type,foaf:Person),
        (y1,foaf:name,?en), (y1,foaf:holdsAccount,F(?an)).
R3-4: zju:emp(?en,?em,?eh), zju:emp_account(?en,?an):-
        (y1,rdf:type,foaf:Person),
        (y1,foaf:name,?en),(y1,foaf:mbox,?em),
        (y1,foaf:schoolHomepage,?eh),
        (y1,foaf:homepage,?eh),(y1,foaf:holdsAccount,F(?an)).
```

Algorithm 1 illustrates the rule generation process. In general, for each view, the algorithm firstly partitions the view body into triple groups. Next,the algorithm generates one class mapping rule for each *triple group*. Sometimes, we need to merge the rules, for example, R3 and R4 are merged into the R3-4 because they are both descriptions for foaf:Person class.

Query Rewriting

In this phase, the Algorithm 2 transforms the input $\mathcal{SHIQ}$ query using the $\mathcal{CMR}$ mapping rules, and outputs a set of valid rewritings that only refer to a set of view heads. The key to the query rewriting is to find out the *applicable class mapping rule*, namely, those mapping rules that are capable of providing (partial) answers, and then replace the query body with the rule heads to yield candidate rewritings.

In details, the rewriting algorithm starts by looking at the body of the query and partitions the body into *triple groups* as Algorithm 1 does. Next, it begins to look for rewritings for each triple group by trying to find an *applicable mapping rules*.

Require: Set of $\mathcal{SHIQ}$-$\mathcal{RDM}$ view $\mathcal{V}$
1: Initialize mapping rules list $\mathcal{M}$,
2: **for all** v in $\mathcal{V}$ **do**
3: Partition the triples in v into triple groups.
4: Let T be the set of triple groups generated for v
5: **for all** triple group t in T **do**
6: create new mapping rule m
7: $\mathfrak{R}(m) = \mathfrak{R}(v)$
8: $\mathfrak{C}(m) = t$
9: add m to $\mathcal{M}$
10: **end for**
11: **end for**
12: Merge rules if they describe the same class.
13: Output mapping rule list $\mathcal{M}$

Algorithm 1: Generating Class Mapping Rules

Definition 6 (Applicable Class Mapping Rule) *Given a triple group t of a query Q, a mapping rule m is an applicable class mapping rule $\mathcal{AM}$ with respect to t, if a.)there is a triple mapping φ that maps every triple in t to a triple in m, and b.)variables in t appearing in the query head also appear in the head of m.*

Definition 7 (Triple Mapping) *Let t_1, t_2 denotes RDF(s) triples and let $\mathsf{Vars}(t_i)$ denotes the set of variables in t_i. t_1 is said to be mapped with t_2* iff *there is a functional variable mapping φ from $\mathsf{Vars}(t_1)$ to $\mathsf{Vars}(t_2)$ such that $t_2 = \varphi(t_1)$.*

Require: Set of mapping rules $\mathcal{M}$, sparql query q
1: Initialize rewriting list Q
2: Let T be the set of triple groups of q
3: Add q to Q
4: **for all** triple group t in T **do**
5: Get all class mapping rules applicable to t, denoted by $\mathcal{AM}$
6: **for all** q in Q **do**
7: remove q from Q
8: **for all** m in $\mathcal{AM}$ **do**
9: Replace t of q with head of m
10: Add q to Q
11: **end for**
12: **end for**
13: **end for**
14: Output rewriting list Q

Algorithm 2: Query Rewriting

If the system finds a $\mathcal{AM}$, it replaces the triple group with the head of the mapping rule, and generates a new partial rewriting. After all triple groups have been replaced, a candidate rewriting is yielded. Figure 2 depicts the rewriting process for query Q_1 using the mapping rules in Example 6. Because of space limitation, only two candidate rewritings are illustrated here.

Given the algorithms, we give an analysis on the complexity of the algorithm. Let n be the number of triple groups in Q_1, let m be the number of mapping rules, it is not difficult to see that in the worst cases the rewriting can be done in time $O(m^n)$. We sketch the analysis as follows. For

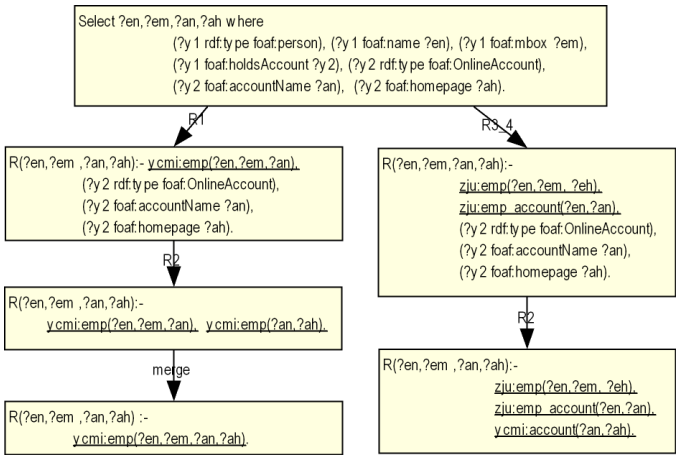

Figure 2: The query rewriting example.

each triple group, the possible number of applicable mapping rules is m. Therefore, in the worst case, there might exist m^n combinations of possible rewritings for Q_1. The worst case experiment in the the evaluation section reflects the correctness of this statement. We note that all rewriting algorithms in LaV settings are limited because a complete algorithm must produce an exponential number of rewritings (Halevy 2001).

Proof of the Correctness of the Algorithm

PROOF: We say that a query $Q1$ is contained in the query $Q2$, denoted by $Q1 \subseteq Q2$, if the answer to $Q1$ is a subset of the answer to $Q2$. To show soundness, we need to show each rewriting R that is obtained by the rewriting algorithm is contained in Q, i.e. $R \subseteq Q$. Since R is unions of conjunctive query rewriting, we show for each conjunctive rewriting $r \in R$, there is a containment mapping from Q to r.

Definition 8 (Containment Mapping) *Let $\mathsf{Vars}(Q)$ denotes the set of variables in Q. A mapping τ from $\mathsf{Vars}(Q_2)$ to $\mathsf{Vars}(Q_1)$ is a containment mapping if*

- *τ maps every triple in the body of Q_2 to a triple in the body of Q_1;*
- *τ maps the head of Q_2 to the head of Q_1.*

Lemma 1 *The Query $Q2$ contains $Q1$ if and only if there is a containment mapping from $Vars(Q2)$ to $Vars(Q1)$.*

Lemma 1 follows from (Chandra A.K. 1977), if we view triples as subgoals.

A conjunctive rewriting has the form: $H(\bar{X}) : -R_1(\bar{X_1}), R_2(\bar{X_2}), ..., R_k(\bar{X_k})$ where $R_i(1 \leq i \leq k)$ is relational predicate. Note, because of the OPTIONAL predicate in Q, it is possible that $\bar{X}$ is not a subset of the union of $R_k(\bar{X_k})$.

Definition 9 (Triple Mapping) *Let t_1, t_2 denotes RDF(s) triples and let $\mathsf{Vars}(t_i)$ denotes the set of variables in t_i. t_1 is said to be mapped with t_2* iff *there is a variable mapping φ from $\mathsf{Vars}(t_1)$ to $\mathsf{Vars}(t_2)$ such that $t_2 = \varphi(t_1)$.*

Definition 10 (Applicable Class Mapping Rule) *Given a triple group t of a query Q, a mapping rule m is an* applicable class mapping rule *$\mathcal{AM}$ with respect to t, if*

- *there is a triple mapping φ that maps every non-optional triple in t to a triple in m;*
- *the variables in t that appears in the query head must also appear in the head of m.*

Given a rewriting r, the expansion of r , denoted by r' is the query in which the relational atoms are replaced by the RDF triples in the body of the class mappings rules that are used to generate r. We denote this rule set as M. To show there is a containment mapping from Q to r ,we would like to show that there exist a containment mapping from Q to r'.

First, the mapping of the head is straitforward. We consider the body: given a non optional triple t in Q, we know that there must exist a mapping rule $m \in M$, which includes $\varphi(t)$, where φ is the triple mapping that maps every non optional triple in a triple group g of query Q to a triple in m. We then know that for each non optional triple t in Q, there must exist a triple t' in r' such that $t' = \varphi(t)$. Thus,we could conclude that there exist a containment mapping from Q to r'. By lemma 1, we conclude $r' \subseteq Q$. Since r is equivalent to r', we continue to get the result of $r \subseteq Q$. Since fore every $r \in R$, $r \subseteq Q$, we finally get $R \subseteq Q$. □

Related Work

Integrating relational databases with semantic web ontologies has always been a hot topic in semantic web communities. Typical works include D2R[2], Virtuoso[3], RDF Gateway, (An Yuan. 2005), etc. In comparison, our mapping system is featured in consideration of the constraint mappings, and the query rewriting algorithm has been incorporated with the description logic reasoning capability which has never been reported in those approaches. Our approach has some similarity to Motik's works (Motik, Horrocks, & Sattler 2007). However, in our system, we first combine the constraint axioms with views to generate new views. Then our query rewriting algorithm is used to answer the query by using new views. So our approach simplify the query process and improve the performance of system.

Our work is also a complement to conventional approaches to integrating relational databases by using description logic such as Information Manifold(Yigal Arens 1996), SIMS (T. Kirk & Srivastava 1995), $\mathcal{DLR}$(Diego Calvanese 1998), $\mathcal{ALCQI}$ (Diego Calvanese 2005), etc.. Besides, the rewriting algorithm absorbs basic ideas from conventional rewriting techniques reported in relational database literatures such as the *bucket algorithm* and the *inver-rule algorithm* (Halevy 2001). Our contribution lies in integrating the constraint mapping information and description logic reasoning capability into query rewriting using views and implementing a practical OWL-to-Relational query rewriting algorithm for semantic web communities.

Conclusion

We report a mapping formalism to bridge the gap between the OWL ontologies and schemata of relational data model (RDB), and present an approach to rewriting $\mathcal{SHIQ}$ conjunctive queries into SQL queries on the basis of the mapping system. In particular, we consider a special mapping called constraint Mapping between RDB integrity constraints and $\mathcal{SHIQ}$ axioms. By including an extra reasoning process over the view, the TBox knowledge and the constraints information are incorporated into the view definitions, thus enabling the query rewriting algorithm to answer more types of queries. Open issues remained include the consideration of the mapping of multi-values dependency of relational data model, and consideration of more expressive description logic languages such as $\mathcal{SHOIN}^+(\mathcal{D})$.

[2] http://sites.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/

[3] http://virtuoso.openlinksw.com

Acknowledgement

This work is supported in part by National Program(No. 2006BAH02A01); National Program(NO.A1420060153); National Program(NO.51306030101).

References

An Yuan., Borgida, A. M. 2005. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *Proceedings of International Semantic Web Conference.*, 6–22.

Chandra A.K., M. P. 1977. Optimal implementation of conjunctive queries in relational databases. In *Proc. Ninth Annual ACM Symposium on Theory of Com- puting*, 77–99.

Diego Calvanese, G. D. G. 1998. Description logic framework for information integration. In *Principles of Knowledge Representation and Reasoning*, 2–13.

Diego Calvanese, G. D. G. 2005. Data integration: A logic-based perspective. *AI Magazine* 25(1):59–70.

Halevy, A. 2001. Answering queries using views: A survey. *Journal of Very Large Database* 10(4):53–67.

Hugo Y.K., K.-H. C. 2007. Alzpharm: Integration of neurodegeneration data using rdf. *BMC Bioinformatics 2007* 8(Suppl 2):S4.

Ian Horrocks, Peter F. Patel-Schneider, F. v. H. 2003. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics* 1(1):"7–26".

Lenzerini, M. 2002. Data integration: a theoretical perspective. In *PODS 2002*, 233–246. New York, NY, USA: ACM Press.

Motik, B.; Horrocks, I.; and Sattler, U. 2007. Bridging the gap between owl and relational databases. In *WWW*, 807–816.

Rachel Pottinger, A. H. 2001. MiniCon: A scalable algorithm for answering queries using views. *VLDB Journal: Very Large Data Bases* 10(2–3):182–198.

T. Kirk, A. Y. Levy, Y. S., and Srivastava, D. 1995. The Information Manifold. In Knoblock, C., and Levy, A., eds., *Information Gathering from Heterogeneous, Distributed Environments*.

Yigal Arens, Craig A. Knoblock, W.-M. S. 1996. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems* 6(2/3):99–130.