P2P Integration of Relational Knowledge Bases

Tadeusz Pankowski
Institute of Control and Information Engineering
Poznań University of Technology
Poznań, Poland
Email: tadeusz.pankowski@put.poznan.pl

***Abstract*—We discuss some strategies of query answering in a Peer-to-Peer (P2P) knowledge integration system. In such a system, a set of autonomous services (peers) manage knowledge bases, which are connected by means of mappings between signatures of these knowledge bases. A query is issued against an arbitrarily chosen peer (a target peer) and is propagated along semantic paths determined by mappings. Next, partial answers are sent back to the target peer. We discuss a strategy of both query propagation and merging partial answers with possibility of discovering some "missing values". The proposed method guarantees improvement of quality of answers (their completeness) and controlling efficiency of merging partial answers by deciding whether it is useful to involve the whole peer's knowledge base in the process of discovering missing values.**

***Keywords*—knowledge bases; data integration; integrity constraints; data exchange; ontology-based data management.**

I. INTRODUCTION

In recent years, we can observe a dynamic development of the Semantic Web technologies and using these technologies to create Web-oriented applications. Semantic Web technologies enable web-wide integration of data coming from various sources. In this way, the Web of Data is created, which can be perceived as a set of interrelated knowledge bases. Extensional layers of these knowledge bases consist of sets of the Resource Description Framework (RDF) triples [20] (or the corresponding Web Ontology Language (OWL) assertions [11]), and intensional layers are sets of axioms (in RDF Schema or OWL). Very often, the knowledge bases expose relational databases – then we can call them *relational knowledge bases* (RKBs). Owners of such RKBs are often interested in making them available to a wide range of users. So, the owners might be interested in including their knowledge resources into knowledge integration or knowledge exchange systems.

Assuming that RKBs independently created resources on the Web of Data, we face the challenging issues of knowledge exchange and knowledge integration across them. In the case of the knowledge exchange, we have to solve problems connected with mappings and restructuring the knowledge (captured by axioms in TBoxes and assertions in ABoxes) [1], and in the case of knowledge integration we have to do with execution and rewriting of queries according to so-called Ontology-Based Data Access (OBDA) paradigm [2].

A. Related work

Some recent results of representing relational databases in the Semantic Web are surveyed in [3] and some solutions were proposed in [4]–[6]. A relationship between relational databases and Description Logics (DLs) knowledge bases has been studied in [7] and [6]. Similarities and differences between databases and knowledge bases, and combining these technologies in data integration activities, has been an important and attractive field of research since many years [7]–[9]. Usually, a knowledge base is founded on one of DL variants [10] or on one of OWL profiles [11]. Formally, a knowledge base is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a set of axioms modeling the intensional knowledge (the TBox axioms), and $\mathcal{A}$ is a set of assertions forming the extensional knowledge (the ABox assertions). In the case of RKBs, $\mathcal{T}$ is divided into two parts, $\mathcal{T} = (\mathcal{S}, \mathcal{C})$, where: $\mathcal{S}$ is a set of axioms treated as *deductive rules* (so-called *standard TBox axioms*), and $\mathcal{C}$ is a set of axioms treated as *checks* (called *integrity constraint* TBox axioms or *ic-axioms*) [7]. Some standards for representing relational databases by means of RDF and OWL have been under development by a special W3C Working Group [5]. Following these RDB-to-RDF methods of mapping, one can create Web of Data repositories consisting of a set of RKBs representing traditional relational databases (RDBs). It is expected that such a representation obeys some requirements concerning *preservation of information and semantics* of the underlying RDBs [4] [6].

One of the most promising integration architectures is the integration in P2P environment [12] [13]. Much work has been done on data integration systems both with a mediated (global) schema and in P2P architecture, where the schema of any peer can play the role of the mediated schema. There is a number of systems built in P2P data integration paradigm [14] (notably Piazza [15], PeerDB [16]). In these research, the focus was on overcoming syntactic heterogeneity and schema mappings were used to specify how data structured under one schema (the source schema) can be transformed into data structured under another schema (the target schema) [17]. A little work has been paid on how schema constraints influence the query propagation.

B. Contribution

In this paper, we will follow the P2P integration architecture and we will assume that the OBDA paradigm can be applied to any peer. Then, the peer's knowledge base plays the role of a reference ontology and mappings (alignments) between ontologies (knowledge bases) are used in query rewriting. The user issues queries against an arbitrarily chosen peer (the target peer) and expects that the answer will include relevant data stored in all P2P-connected data sources. The data sources are related by means of schema mappings. A query must be propagated to all peers in the system along semantic paths determined by mappings and reformulated accordingly.

 ISBN: 978-1-61208-346-9

The partial answers must be merged and sent back to the calling peer.

In this paper, we will focus on strategies of query propagation and query answering in a P2P knowledge integration system. The aim is to obtain an answer with a maximal information contents. We will show that this contents depends on the way the query is propagated and on the way in which the partial answers are being merged. The important role plays the process of discovering *missing values*, i.e., values denoted by labeled nulls. Missing values can be sometimes discovered – either by merging only partial answers or in result of involving also the whole local RKB. In the latter, the cost of computation can be high, so the decision what way of merging should be applied is significant.

We formulate and prove a necessary condition (Proposition 4.2 in Section IV) stating when it is useful to discover missing values by referring to the whole local RKB, and when it is pointless. We will discuss a strategy of query propagation and the method of merging partial answers with possibility of discovering some missing values. The proposed method guarantees improvement of quality of answers (their completeness) and controlling efficiency of merging partial queries by deciding whether it is useful to involve a whole peer's knowledge base in the process of discovering missing values. We shortly show how the issues under consideration have been implemented in SixP2P (*Semantic Integration in P2P environment*) system [18] [19].

The structure of the paper is as follows. In Section II, we introduce a running example consisting of relational databases (RDBs), we study some query propagation strategies and merging partial answers. In Section III, we discuss a way of representing RDBs by means of RKBs. The P2P knowledge integration is proposed in Section IV. Section V summarizes and concludes the paper.

II. MOTIVATION SCENARIO

In this section, we motivate our research. We will start with three relational databases (Figure 1) as information resources stored in three peers forming a P2P data integration system (Figure 2). We discuss how a sample query can be propagated and answered in the system, and how these influence the contents of the answer to the query.

Without loss of generality we will assume that names and attributes of tables in databases are pair-wise disjoint. Integrity constraints in DB_1 (analogously in DB_2 and DB_3) are:

- $PKey(Paper1, PapId1)$ – $PapId1$ is the primary key in $Paper1$, i.e., values in $PapId1$ are both unique and not-null;
- $unique(Paper1, Title1)$ – not-null values in $Title1$ uniquely identifies tuples in $Paper1$;
- $FKey(Author1, APapId1, Paper1, PapId1)$ – $APapId1$ in $Author1$ is a foreign key referencing to the column $PapId1$ in $Paper1$.

The databases considered in Figure 1 can be stored in peers constituting a data integration system depicted in Figure 2.

DB_1 :

$Paper1$

$PapId1$	$Title1$	$Year1$

$Author1$

$Name1$	$Univ1$	$APapId1$

DB_2 :

$Paper2$

$PapId2$	$Title2$
p1	KBs

$Author2$

$Name2$	$Univ2$	$APapId2$
John	NY	p1
Ann	NULL	p1

DB_3 :

$Paper3$

$PapId3$	$Title3$	$Year3$
p1	KBs	2014

$Author3$

$Name3$	$APapId3$
Ann	p1

Figure 1. Databases DB_1 (with empty instance), DB_2 and DB_3

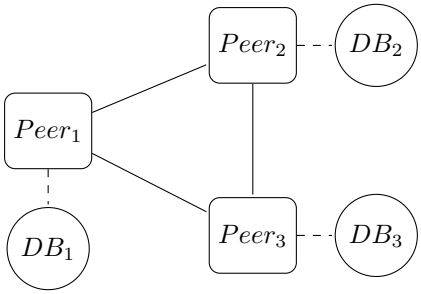

Figure 2. A sample P2P integration architecture with three peers and three local databases

Now, let us consider a query q_1 issued against DB_1 on $Peer1$. Informally, the meaning of the query is:

"Give all information about John."

The query can be answered using different strategies. We will consider three strategies depicted in Figure 3.

1) In strategy (1), see Figure 3(1), q_1 is issued to DB_1 (at $Peer_1$). $Peer_1$ rewrites q_1 to queries q_{12} and q_{13} against, respectively, DB_2 and DB_3 and sends them to $Peer_2$ and $Peer_3$, respectively. Next, the answer $q_1(DB_1)$ is obtained and the peer waits for answers $q_{12}(DB_2)$ and $q_{13}(DB_3)$. After receiving all expected answers, the peer merges them producing the final answer. In this case we have:
$$q_1(DB_1) = q_{13}(DB_3) = \emptyset,$$
$$q_{12}(DB_2) = \{Paper2(p1, KBs), Author2(John, NY, p1)\}.$$
2) In strategy (2), see Figure 3(2), q_1, q_{12}, and q_{13}, are sent to, respectively, $Peer_1$, $Peer_2$, and $Peer_3$, as in the strategy (1). $Peer_2$ rewrites q_{12} to q_{123} and sends it to $Peer_3$. Now, $Peer_3$ has to answer two queries, q_{13} and q_{123}. These queries are not identical because

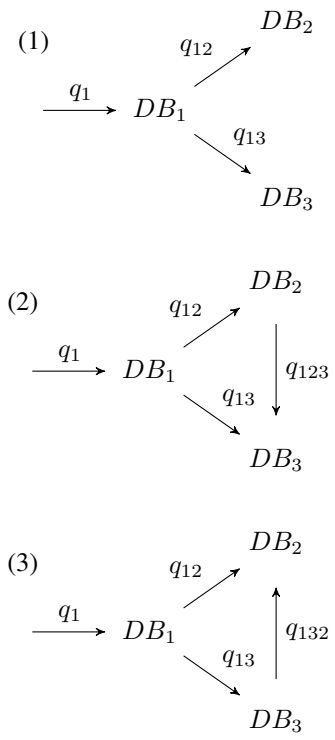

Figure 3. Query propagation strategies

q_{13} expects information about $Year3$ whereas q_{123} does not, since the property $Year$ is not in its area of interest. Answers $q_{13}(DB_3)$ and $q_{123}(DB_3)$ are expected to be sent to, respectively, $Peer_1$ and $Peer_2$. $Peer_2$ merges $q_{123}(DB_3)$ with $q_{12}(DB_2)$ and the result returns to $Peer_1$. Finally, $Peer_1$ merges all received data and produces the final result. In this case we have:

$$q_1(DB_1) = q_{13}(DB_3) = q_{123}(DB_3) = \emptyset,$$
$$q_{12}(DB_2) = \{Paper2(p1, KBs), Author2(John, NY, p1)\}.$$

3) Strategy (3), see Figure 3(3), is similar to strategy (2), but now, $Peer_3$ rewrites q_{13} to q_{132} and sends it to $Peer_2$. We have the following answers:

$$q_1(DB_1) = q_{13}(DB_3) = \emptyset,$$
$$q_{12}(DB_2) = \{Paper2(p1, KBs), Author2(John, NY, p1)\},$$
$$q_{132}(DB_2) = \{Paper2(p1, KBs), Author2(John, , p1)\}.$$

The difference between $q_{12}(DB_2)$ and $q_{132}(DB_2)$ follows from the fact that $Peer_3$ does not ask about university of "John".

Now, we will consider some possible ways for merging answers and discovering missing values in the process of data integration. We will use the following two operators:

- $\otimes_i$ – a binary operator that: restructures its operands (if necessary) to the structure of DB_i and merges these operands;
- $\oplus_i$ – a unary operator that discovers missing values in its operand using DB_i.

Merging in strategy (1):

$$q_1(DB_1) \otimes_1 q_{12}(DB_2) \otimes_1 q_{13}(DB_3) = \{Paper1(p1, KBs, \texttt{NULL}), Author1(John, NY, p1)\} \quad (1)$$

Merging in strategy (2). There are two possible merges:

(2a) a partial merge

$$q_1(DB_1) \otimes_1 q_{13}(DB_3) \otimes_1 (q_{12}(DB_2) \otimes_2 q_{123}(DB_3)) = \{Paper1(p1, KBs, \texttt{NULL}), Author1(John, NY, p1)\} \quad (2)$$

(2b) a total merge

$$q_1(DB_1) \otimes_1 q_{13}(DB_3) \otimes_1 (\oplus_2(q_{12}(DB_2) \otimes_2 q_{123}(DB_3))) = \{Paper1(p1, KBs, \texttt{NULL}), Author1(John, NY, p1)\} \quad (3)$$

Merging in strategy (3). Again, two merges are possible:

(3a) a partial merge

$$q_1(DB_1) \otimes_1 q_{12}(DB_2) \otimes_1 (q_{132}(DB_2) \otimes_3 q_{13}(DB_3)) = \{Paper1(p1, KBs, \texttt{NULL}), Author1(John, NY, p1)\} \quad (4)$$

(3b) a total merge

$$q_1(DB_1) \otimes_1 q_{12}(DB_2) \otimes_1 (\oplus_3(q_{132}(DB_2) \otimes_3 q_{13}(DB_3))) = \{Paper1(p1, KBs, 2014), Author1(John, NY, p1)\} \quad (5)$$

We see that merging strategies (1), (2a), (2b) and (3a) produce the same result. However, the strategy (3b) gives more information than the other strategies. It is so, since in (3b) the missing value of $Year1$ (the year of publication of the paper KBs) has been discovered (inferred). It is possible, because there is a functional dependency between titles and years of papers.

III. Relational DB vs relational KB

Relationships between RDBs and RKBs can be considered from the following two points of view.

1. *RDB-to-RKB transformation*. RDBs can be naturally represented in the Semantic Web by means of RDF triples or OWL specification [5]. Then, the instance of an RDB is represented by an ABox ($\mathcal{A}$), and the structure, properties and integrity constraints by a TBox (devided into a set of deductive axioms, $\mathcal{S}$, and a set of ic-axioms, $\mathcal{C}$). Then we obtain an RKB $\mathcal{K} = (\mathcal{S}, \mathcal{C}, \mathcal{A})$ [6].

2. *Checking properties of RKB*. For a given set of RDF triples or OWL assertions, treated as an ABox $\mathcal{A}$, and for a given RKB schema $\mathcal{T} = (\mathcal{S}, \mathcal{C})$, check whether the tuple $(\mathcal{S}, \mathcal{C}, \mathcal{A})$ is a consistent RKB.

Let $\mathbf{C} = \{C_1, ..., C_k\}$ and $\mathbf{P} = \{P_1, ..., P_p\}$ be sets of (names of), respectively, *classes* and *properties*. The set $\Sigma = \mathbf{C} \cup \mathbf{P}$ is then referred to as a *signature* of a knowledge base. We say that an RKB $\mathcal{K} = (\mathcal{S}, \mathcal{C}, \mathcal{A})$ has the signature Σ (or $\mathcal{K}$ is over Σ) if all classes and properties occurring in $\mathcal{K}$ are in Σ.

 ISBN: 978-1-61208-346-9

A. Relational knowledge bases

Let Δ_{Cons} be a set of *constants*, and Δ_{Var} be a set of *labeled nulls* or *variables*. We assume that for constants the Unique Name Assumption (UNA) holds while for labeled nulls the UNA does not hold [8]. Constants will be denoted by a, b, c (possibly with subscripts), and labeled nulls by x, y, z (possibly with subscripts), by v (possibly with subscripts) will be denoted elements from $\Delta_{Cons} \cup \Delta_{Var}$. The satisfaction of UNA means that different constants always denote different objects, i.e., the equality $a = b$, for $a, b \in \Delta_{Cons}$, is always false. In contrast, $x = y$ for $x, y \in \Delta_{Var}$ may be true if an interpretation assigns the same object to them. Similarly, $x = a$ can be true when x is interpreted as a. (For the precise definition of interpretation, see [10].)

Further on, we will write ABox assertions in a (simplified) RDF-notation [20]. In particular: $Triple(x, \text{rdf:type}, C)$ or $(x, \text{rdf:type}, C)$ corresponds to OWL class assertion $C(x)$; and $Triple(x, P, v)$ or (x, P, v) corresponds to OWL property assertion $P(x, v)$. If clear from the context, the predicate name $Triple$ will be omitted.

We assume that the following rules must always hold for any RKB. We will call them *general RKB-rules*.

- Classes are subsets of the set of labeled nulls, i.e., for each $C \in \mathbf{C}$, $C \sqsubseteq \Delta_{Var}$.[1]
- Domains of properties are subsets of the set of labeled nulls, i.e., for each $P \in \mathbf{P}$, $dom(P) \sqsubseteq \Delta_{Var}$.
- Ranges of properties are subsets of the set of constants or labeled nulls, i.e., for each $P \in \mathbf{P}$, $rng(P) \sqsubseteq \Delta_{Cons} \cup \Delta_{Var}$.
- Any property is a function, i.e., for each $P \in \mathbf{P}$, the specification $(\texttt{funct}\ P) \in \mathcal{S}$ holds, which means that

$$(x, P, v_1) \wedge (x, P, v_2) \Rightarrow v_1 = v_2.$$

There are also so-called *specific RKB-rules*, which are used to characterize RKBs. These rules correspond to integrity constraints in relational databases. We restrict ourselves to key and referential constraints defined only on singletons of columns. The set of rules is divided into a set of *deductive rules* ($\mathcal{S}$), and a set of *integrity constraints* or *check rules* ($\mathcal{C}$). We assume that $(\texttt{funct}\ P) \in \mathcal{S}$. The rest of general RKB-rules belongs to $\mathcal{C}$. The set of specific RKB-rules includes the following classes of rules (TBox axioms):

1) $dom(P) \sqsubseteq C \in \mathcal{S}$ – the domain of a property P is a subset of a class C, i.e.,

$$(x, P, v) \Rightarrow (x, \text{rdf:type}, C).$$

2) $(\texttt{funct}\ P^-) \in \mathcal{S}$ – the inversion of a property P is a function, i.e., values of P uniquely identify objects in the domain of P

$$(x_1, P, v) \wedge (x_2, P, v) \Rightarrow x_1 = x_2.$$

3) $C \sqsubseteq dom(P) \in \mathcal{C}$ – a class C is a subset of the domain of P, i.e., P is a total (not-null) on C

$$(x, \text{rdf:type}, C) \Rightarrow \exists v.(x, P, v).$$

4) $rng(P) \sqsubseteq C \in \mathcal{C}$ – the range of P is a subset of a class C, i.e., P references objects in C

$$(x, P, x') \Rightarrow (x', \text{rdf:type}, C).$$

B. Transformation of RDBs to RKBs

Each RDB DB_i, depicted in Figure 1, can be translated into an RKB $\mathcal{K} = (\mathcal{S}_i, \mathcal{C}_i, \mathcal{A}_i)$, where $\mathcal{S}_i$ and $\mathcal{C}_i$, for $i = 1, 2, 3$, can be created from $\mathcal{S}$ and $\mathcal{C}$ in Figure 4 by appropriate extension of the property names and class names with postfixes '1', '2', and '3'. Additionally, in $\mathcal{S}_2$ do not occur rules involving $Year2$, and in $\mathcal{S}_3$ are not rules containing $Univ3$.

$$\begin{aligned}
\mathcal{S} = \{&dom(PapId) \sqsubseteq Paper,\\
&dom(Title) \sqsubseteq Paper,\\
&dom(Year) \sqsubseteq Paper,\\
&dom(Name) \sqsubseteq Author,\\
&dom(Univ) \sqsubseteq Author,\\
&dom(APapId) \sqsubseteq Author,\\
&(\texttt{funct}\ PapId^-), (\texttt{funct}\ Title^-)\}.\\
\mathcal{C} = \{&Paper \sqsubseteq dom(PapId), Paper \sqsubseteq dom(Year),\\
&Author \sqsubseteq dom(Name), rng(APapId) = Paper\}.
\end{aligned}$$

Figure 4. Standard TBox axioms ($\mathcal{S}$) and TBox ic-axioms ($\mathcal{C}$) corresponding to schemas of RDBs in Figure 1. Names should be appropriately extended with postfixes '1', '2', and '3'.

Then we obtain the following RKBs, $\mathcal{K}_i = \tau(DB_i), i = 1, 2, 3$, where τ denotes the translation operator from RDBs into RKBs:

1) $\tau(DB_1) = \mathcal{K}_1 = (\mathcal{S}_1, \mathcal{C}_1, \mathcal{A}_1)$, where:

$$\mathcal{A}_1 = \emptyset.$$

2) $\tau(DB_2) = \mathcal{K}_2 = (\mathcal{S}_2, \mathcal{C}_2, \mathcal{A}_2)$, where:

$$\begin{aligned}
\mathcal{A}_2 = \{&(x_1, PapId2, p1),\\
&(x_1, Title2, KBs),\\
&(x_2, Name2, John),\\
&(x_2, Univ2, NY),\\
&(x_2, APapId2, x_1),\\
&(x_3, Name2, Ann),\\
&(x_3, APapId2, x_1)\}.
\end{aligned}$$

3) $\tau(DB_3) = \mathcal{K}_3 = (\mathcal{S}_3, \mathcal{C}_3, \mathcal{A}_3)$, where:

$$\begin{aligned}
\mathcal{A}_3 = \{&(y_1, PapId3, p1),\\
&(y_1, Title3, KBs),\\
&(y_1, Year3, 2014),\\
&(y_2, Name3, Ann),\\
&(y_2, APapId3, y_1)\}.
\end{aligned}$$

A set $\mathcal{A}$ of ABox assertions arises from an instance of relational database in the result of the translation performed by Algorithm 1.

In Algorithm 1, for each relation symbol R and each attribute $A \in att(R)$:

[1] We distinguish between the semantic notion of *inclusion* ($\subseteq$), and the syntactic notion of *subsumption* ($\sqsubseteq$). Then $A \sqsubseteq B$ iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$, where $\mathcal{I}$ is an interpretation function, see [10].

 ISBN: 978-1-61208-346-9

Algorithm 1 Creating ABox assertions

Input: An RDB $(\mathbf{R}, IC, I)$, and an empty ABox $\mathcal{A}$.
Output: ABox assertions in $\mathcal{A}$ representing I.
for each $R(t) \in I$
  $x :=$ a fresh labeled null in Δ_{Var}
  $U_{R,t} := \{A \in att(R) \mid t.A \neq \texttt{NULL}\}$
  if $U_{R,t} = \emptyset$ **then**
    $\mathcal{A} := \mathcal{A} \cup \{(x, \text{rdf:type}, C_R)\}$
  else
  for each $A \in U_{R,t}$
    if $FKey(R, A, R', A') \in IC$ **and**
    exists x' **such that** $(x', P_{A'}, t.A) \in \mathcal{A}$ **then**
      $\mathcal{A} := \mathcal{A} \cup \{(x, P_A, x')\}$
    else
      $\mathcal{A} := \mathcal{A} \cup \{(x, P_A, t.A)\}$
end

a) a fresh labeled null x is selected from Δ_{Var} for each tuple t in an instance of R;
b) for each $A \in att(R)$ if $t.A = a \neq \texttt{NULL}$ and A is not a foreign key in R, the triple (x, P_A, a) is inserted into $\mathcal{A}$; so, fields with NULL values are omitted;
c) if A is a foreign key in R, i.e., $FKey(R, A, R', A') \in IC$, and $t.A = a$, then the triple (x, P_A, x') is inserted into $\mathcal{A}$, where x' is such that $(x', P_{A'}, a) \in \mathcal{A}$;
d) if for each attribute $A \in att(R)$, $t.A = \texttt{NULL}$ then $(x, \text{rdf:type}, C_R)$ is inserted into $\mathcal{A}$.

IV. P2P KNOWLEDGE INTEGRATION

Processing a query in a peer-to-peer environment is presented in Algorithm 2. In general, the algorithm is self-explaining. However, some procedures will be discussed deeply later on, namely:

- $margeAnswers(curPeer, query)$ – merge all answers to $query$ gathered in $curPeer$;
- $canDiscover(curPeer, query)$ – decide whether some missing values in the answer to $query$ can be discovered using the whole RKB stored in $curPeer$;
- $discoverUsingRKB(curPeer, query)$ – discover missing values in the answer to $query$, using the whole RKB in $curPeer$.

We consider conjunctive queries, which are conjunctions of equalities of the form $P = a$, where P is a property and a is a constant.

By $Triple(x, p, v)$ we denote a triple that either is in $\mathcal{A}$ or can be deduced from $\mathcal{A}$ by means of $\mathcal{S}$ (in the current peer's RKB, $\mathcal{K} = (\mathcal{S}, \mathcal{C}, \mathcal{A})$). An answer to $P = a$ consists of all triples determined by the following recursive datalog program:

$$Answer(x, p, v) \leftarrow Triple(x, p, v), p = \text{"}P\text{"}, v = \text{"}a\text{"}$$
$$Answer(x, p, v) \leftarrow Answer(x, \_, \_), Triple(x, p, v)$$
$$Answer(x, p, v) \leftarrow Answer(\_, \_, x), Triple(x, p, v)$$

The answer encompasses all triples, which describe an object x with the value a of property P, all other properties of x, and triples connected with x by means of references.

Further on, we will use i instead of $Peer_i$, and we assume the following denotations:

Algorithm 2 Processing a query in a peer

Input: Set of peers with local RKBs. A peer is connected with its partners by means of mappings between signatures of RKBs.
$query$ - a query against a current peer.
Output: Answer to the $query$ containing local answer and answers returned by all peer's partners.

for each $srcPeer$ **in** $partnersOf(curPeer)$ {
  $query = rewriteQuery(curPeer, srcPeer, query)$;
    ▷ $query$ is rewritten using the mapping from $curPeer$ to $srcPeer$
  $sendQuery(curPeer, srcPeer, query)$;
    ▷ $query$ is sent from $curPeer$ to $srcPeer$
}
for each $srcPeer$ **in** $partnersOf(curPeer)$ {
  $answer = answer \cup$
    $transformAnswer(curPeer, srcPeer, query)$;
    ▷ $answer$ to $query$ from $srcPeer$ is gathered in $curPeer$
}
$answer = margeAnswers(curPeer, query)$;
    ▷ all answers to $query$ are merged in $curPeer$
if $canDiscover(curPeer, query)$ **then** {
  $answer = discoverUsingRKB(curPeer, query)$;
    ▷ $curPeer$'s RKB is used to discover missing values
    ▷ in the answer to $query$
}

1) $\{i_1, \ldots, i_n\}$ – a set of partners of a peer i;
2) $\mathcal{M}_{k,i}$ – a mapping between signatures Σ_k and Σ_i of RKBs stored in peers k and i, respectively;
3) $\mathcal{M}_{k,i}(ans_k(q))$ – the transformation of an answer $ans_k(q)$ specified by a mapping $\mathcal{M}_{k,i}$.

Then the answer $ans_i(q)$ is obtained in the following steps:

Step 1. The answer $q(i)$, i.e., the local answer to q w.r.t. the current RKB is returned

$$ans_i(q) = q(i).$$

Step 2. Answers of all i's partners are added to $ans_i(q)$

$$ans_i(q) = ans_i(q) \cup \{\mathcal{M}_{k,i}(ans_k(q)) \mid k \in \{i_1, \ldots, i_n\}\}.$$

Step 3. Axioms in $\mathcal{S}_i \cup \mathcal{C}_i$ are used to deduce all possible facts in $ans_i(q)$

$$ans_i(q) = ans_i(q) \cup \mathcal{S}_i \cup \mathcal{C}_i = \{(x, p, v) \mid ans_i(q) \cup \mathcal{S}_i \cup \mathcal{C}_i \vdash (x, p, v)\}.$$

Let us focus on the merge operation in $Peer_3$ – see the merging strategy (5) in Section II applied to RKBs – we have:

Step 1 : $ans_3(q) = \emptyset$;
Step 2 : $ans_3(q) = ans_3(q) \cup \mathcal{M}_{23}(ans_2(q)) =$
  $\{(x_1, PapId3, p1), (x_1, Title3, KBs),$
  $(x_2, Name3, John), (x_2, APapId3, x_1)\}$;
Step 3 : $ans_3(q) = ans_3(q) \cup \mathcal{S}_3 \cup \mathcal{C}_3 =$
  $\{(x_1, PapId3, p1), (x_1, Title3, KBs),$
  $(x_2, Name3, John), (x_2, APapId3, x_1),$
  (a) $(x_1, \text{rdf:type}, Paper3),$
  (b) $(x_2, \text{rdf:type}, Author3),$
  (c) $(x_1, Year3, x_3)\}$.

Triples (a) and (b) were deduced using $dom(Title3) \sqsubseteq Paper3 \in \mathcal{S}_3$, and $dom(Name3) \sqsubseteq Author3 \in \mathcal{S}_3$. Triple

(c) was deduced by application of the axiom $Paper3 \sqsubseteq dom(Year3) \in \mathcal{C}_3$ to the triple $(x_1, \text{rdf:type}, Paper3)$. Note that x_3 is a "fresh" labeled null in Δ_{Var}.

Now, the question is whether the missing value represented by x_3 can be discovered referring to the whole RKB $\mathcal{K}_3$. To this order we can proceed as follows:

$$\{(x_1, Title3, KBs), (x_1, Year3, x_3)\} \subseteq ans_3(q),$$
$$\{(y_1, Title3, KBs), (y_1, Year3, 2014)\} \subseteq \mathcal{A}_3.$$

Since $Title3$ is unique in $\mathcal{A}_3 \cup ans_3(q)$, which is stated by the axiom $(\texttt{funct}\ Title3^-) \in \mathcal{S}_3$, we have $x_1 = y_1$. Now we can apply the axiom $(\texttt{funct}\ Year3) \in \mathcal{S}_3$ to the set

$$\{(y_1, Year3, 2014), (y_1, Year3, x_3)\} \subseteq \mathcal{A}_3 \cup ans_3(q).$$

From functionality of *Year*3, we have $x_3 = 2014$. In this way the missing value represented by x_3 has been discovered.

Now, we can describe the process of discovering missing values more precisely. Proposition 4.2 formulates the necessary condition for the possibility of success in the discovery process.

Definition 4.1: Let a triple (x, P, x') be in an a set $ans(q)$ of triples constituting an answer to a query q. We say that x' is a *missing value* of P if there is not any triple in $ans(q)$ whose first component (the subject) is x'.

It means that a labeled null x' in a triple (x, P, x') represents a *missing value* if the property P is not interpreted as a reference.

Proposition 4.2: Let P occur in a query q issued against a peer, $\mathcal{K} = (\mathcal{S}, \mathcal{C}, \mathcal{A})$ be an RKB in this peer, and $ans(q)$ be a result of merging all partial answers returned to this peer. Let a triple $(x, P_2, x') \in ans(q)$, where x' is a missing value of P_2. The necessary condition for discovering the value of x' is:

(c1) there is a triple $(x, P_1, a) \in ans(q)$, $P_1 \neq P$, and
(c2) P_1 is unique in its domain, i.e., $(\texttt{funct}\ P_1^-) \in \mathcal{S}$.

Proof: Note that the given condition is a necessary condition.

1) Let (c1) and (c2) be true. Assume that the following two triples: (y, P_1, a) and (y, P_2, b) are in $\mathcal{A}$. Then, from the uniqueness of P_1 it follows that $y = x$, and from functionality of P_2 w obtain $x' = b$. So, b is the discovered value of x'.
2) Let $P_1 = P$ and (c2) hold. Because $ans(q)$ contains also the answer to q returned by $\mathcal{K}$ then the triple (y, P_1, a) (mentioned in 1)) occurs either in both $\mathcal{A}$ and $ans(q)$, or in neither of them. Thus, to discover the missing value of x' we can restrict ourselves only to $ans(q)$.
3) Now, let (c1) holds and (c2) is not true. Then the equality $y = x$, considered in 1) can not be deduced, and the discovering process fails.

Thus, the thesis of the proposition holds. ■

V. Conclusions and Future Work

We discussed the problem of aswering queries issued in a knowledge integration systems in P2P architecture. In such a system, there are many autonomous services (peers), which collaborates in the process of producing answers to queries. The system is flexible and peers can enter and leave the system dynamically. A peer has a knowledge base and while entering the system it establishes semantic relationships between the signature of its knowledge base and signatures of some knowledge bases already belonging to the system (its partners). A query posed against a peer is propagated to its partners along semantic paths defined by mapping, those partners propagate the query to their partners, etc. Answers to the query flow back in the opposite directions. The target peer merge the answers producing the expected answer. In this merge either only answers are taken into account or also the whole RKB stored in the peer can be involved. This influence the quality (completeness) of the answer as well as the efficiency of the query answering process. We have shown how one can control these factors of the knowledge integration system. In particular, we discussed when the involvement of the whole RKB is useful from the so-called *"discovery of missing values"* point of view. The proper decision significantly influence both the quality of the answer as well as efficiency of processing. The discussed strategy was inspired by the SixP2P system originally designed for integrating XML data [18] [19]. The future extension of the implementation is intended to capture also semantic-oriented repositories, such as relational knowledge bases organized in a form of RDF triples or OWL specifications.

Acknowledgment

This research has been supported by Polish Ministry of Science and Higher Education under grant 04/45/DSPB/0122.

References

[1] M. Arenas, E. Botoeva, and D. Calvanese, "Knowledge base exchange," in 24th International Workshop on Description Logics (DL 2011), 2011, pp. 1–11.

[2] D. Calvanese, G. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family," J. Autom. Reason., vol. 39, no. 3, 2007, pp. 385–429.

[3] J. Sequeda, S. H. Tirmizi, Ó. Corcho, and D. P. Miranker, "Survey of directly mapping SQL databases to the Semantic Web," Knowledge Eng. Review, vol. 26, no. 4, 2011, pp. 445–486.

[4] J. Sequeda, M. Arenas, and D. P. Miranker, "On Directly Mapping Relational Databases to RDF and OWL (Extended Version)," CoRR, vol. abs/1202.3667, 2012, pp. 1–17.

[5] M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda, "A Direct Mapping of Relational Data to RDF," 2012, http://www.w3.org/TR/rdb-direct-mapping, [retrieved: May, 2014].

[6] T. Pankowski, "Reasoning About Consistency Of Relational Knowledge Bases," in 8th International Multi-Conference on Computing in the Global Information Technology, ICCGI 2013, July 21 - 26, 2013 - Nice, France. IARIA, 2013, pp. 283–288.

[7] B. Motik, I. Horrocks, and U. Sattler, "Bridging the gap between OWL and relational databases," Journal of Web Semantics, vol. 7, no. 2, 2009, pp. 74–89.

[8] S. Abiteboul, R. Hull, and V. Vianu, Foundations of Databases. Reading, Massachusetts: Addison-Wesley, 1995.

[9] R. Reiter, "Towards a logical reconstruction of relational database theory," in On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and programming Languages, 1982, pp. 191–233.

[10] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Petel-Schneider, Eds., The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

 ISBN: 978-1-61208-346-9

[11] OWL 2 Web Ontology Language Profiles, 2009, www.w3.org/TR/owl2-profiles, [retrieved: May, 2014].

[12] D. Calvanese, E. Damaggio, G. D. Giacomo, M. Lenzerini, and R. Rosati, "Semantic data integration in P2P systems," in DBISP2P-2003, LNCS 2944, Springer, 2004, pp. 77–90.

[13] F. Buccafurri and G. Lax, "Enabling Selective Flooding to Reduce P2P Traffic," in OTM Conferences, LNCS 4803, Springer, 2007, pp. 188–205.

[14] G. Koloniari and E. Pitoura, "Peer-to-peer management of XML data: issues and research challenges," SIGMOD Record, vol. 34, no. 2, 2005, pp. 6–17.

[15] I. Tatarinov, et al., "The Piazza peer data management project," SIGMOD Record, vol. 32, no. 3, 2003, pp. 47–52.

[16] B. C. Ooi, Y. Shu, , and K.-L. Tan, "Relational data sharing in peer-based data management systems," SIGMOD Record, vol. 32, no. 3, 2003, pp. 59–64.

[17] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan, "Peer data exchange," ACM Trans. Database Syst, vol. 31, no. 4, 2006, pp. 1454–1498.

[18] G. Brzykcy, J. Bartoszek, and T. Pankowski, "Schema Mappings and Agents' Actions in P2P Data Integration System," Journal of Universal Computer Science, vol. 14, no. 7, 2008, pp. 1048–1060.

[19] T. Pankowski, "Query propagation in a P2P data integration system in the presence of schema constraints," in Data Management in Grid and P2P Systems DEXA/Globe'08, LNCS 5187, 2008, pp. 46–57.

[20] Resource Description Framework (RDF) Model and Syntax Specification, 1999, www.w3.org/TR/PR-rdf-syntax/, [retrieved: May, 2014].

 ISBN: 978-1-61208-346-9