Faceted Queries in Ontology-based Data Integration

Tadeusz Pankowski
Institute of Control and Information Engineering, Poznań University of Technology, Poznań, Poland

Keywords: Data Integration, Faceted Queries, Ontology, RDF Data.

Abstract: The aim of using ontology-based data integration is to provide users with a unified view, in a form of a global application domain ontology, over a multitude of data sources. The terminological component of this ontology is then presented as the global schema of the system and is used as the reference model for formulating queries. The extensional knowledge consists of RDF data sets (graphs) stored in local databases. In such scenario, a faceted query interface is a desired solution for end-user data access. Then there is a need for effective query answering utilizing both extensional and intentional knowledge representation. In this paper, we propose and discuss a possible solution to this issue. We show how a class of deductive rules, in particular Datalog rules and rules defining functionality, can be incorporated in the process of ontology-enhanced query answering in ontology-based data integration systems.

1 INTRODUCTION

Data integration systems provide users with a uniform view over a multitude of heterogeneous data sources. This uniform view has a form of a *global schema*, which realize so called global-as-view (GAV) paradigm (Lenzerini, 2002), (Ullman, 1997). The data is stored in data sources in their own schemas. The global schema frees users from having to locate the sources relevant to their queries. In order to answer queries formulated against the global schema, the system provides the semantic mappings between the global schema and the local (source) schemas (Halevy et al., 2006), (Calì et al., 2004), (Fagin et al., 2009), (Bernstein and Haas, 2008).

Currently, a broad class of data integration systems uses ontologies as global schemas, which led to emergence of *ontology-based data integration* (OBDI) and to *ontology-based data access* (OBDA) (Cruz and Xiao, 2009), (Calvanese et al., 2010), (Das et al., 2004), (Eklund et al., 2004), (Calvanese et al., 2007a), (Skjæveland et al., 2015).

Now, the most popular means to specify and query ontologies are OWL (OWL 2 Web Ontology Language Profiles, 2009), RDF (Resource Description Framework (RDF) Model and Syntax Specification, 1999) and SPARQL (SPARQL Query Language for RDF, 2008). OWL provides a method to formalize a domain by defining classes and properties of those classes (by means of *rules* or *axioms*), and to define individuals and assert properties about them (by means of *facts* or *assertions*). OWL is based on description logic (Baader et al., 2003). In practice, ontology rules are usually written in a form of first order language (FOL) formulas, and facts are usually defined by means of RDF graphs or FOL sentences. A standard language to formulate queries over ontologies is SPARQL. SPARQL, however, is not suitable query language for end-users. Instead, so called *faceted search* is used for end-user data access (Yee et al., 2003), (Oren et al., 2006), (Hahn et al., 2010).

In OBDI systems, an ontology is divided into two components: *terminological component* (TBox) consisting of *signature* (a set of unary predicate names (classes) and binary predicate names (properties)), and *rules* (axioms); and *assertional component* consisting of a set of *facts* (assertions). The terminological component forms the *global schema* of OBDI, and the assertional component consists of a set of local databases. Any local database state is a set of RDF data, which can be represented as an RDF graph, so we call it a *graph database*. The set of RDF graphs forms *extensional* knowledge. The set of rules in global schema constitutes the *intentional* knowledge about the application domain, and substantially enrich the extensional knowledge. The challenging issue in OBDI is to take into account both the extensional and intentional knowledge while answering queries. Data in different local databases can complement each other and can overlap. However, we as-

Pankowski, T.
Faceted Queries in Ontology-based Data Integration.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 1*, pages 150-157
ISBN: 978-989-758-187-8

sume that they are consistent with the global schema and do not contradict one another.

In this paper, we propose a method for query answering in OBDI systems in the situation when queries are formulated against the global schema as *faceted queries*. To create the answer, the service uses relevant data from all local databases as well as data necessary in reasoning procedures implied by ontology rules.

The main contribution of the paper, is the proposal of an algorithm for extending the query graph (created from a faceted query) with edges implied by relevant deductive rules. The set of considered ontology rules includes so called Datalog rules and rules specifying functionality of binary predicates and their inversions (key properties). We also show how the extended query graph (a *global query pattern*) can be used to merge local answers by means of a chase procedure.

The structure of the paper is as follows. Some preliminaries concerning graph databases, ontologies and queries, are reviewed in Section 2. Faceted queries are defined in Section 3, and their formal semantics, understood as first order open formulas, is given. In Section 4 we characterize ontology-based data integration, and describe architecture of an OBDI system. Also the running example is introduced. The process of answering faceted queries is described in Section 5. We propose an algorithm for creating global, ontology-enhanced query graph, and its usage in merging local answers and obtaining the final result. Section 6 concludes the paper.

2 PRELIMINARIES

A *graph database* is a finite, edge-labeled and directed graph (Barceló and Fontaine, 2015). Formally, let the following sets be infinite and pairwise disjoint: Const – a set of *constants*, $\mathsf{LabNull}$ – a set of *labeled nulls* (treated as variables), UP – a set of *unary predicates*, BP – set of *binary predicates*. Additionally, we assume that type and $\approx$ are distinguished binary predicates in BP. A *signature* Σ is a finite subset of $\mathsf{UP} \cup \mathsf{BP}$.

A *graph database* (or *RDF graph*) $G = (V, E)$ with signature Σ consists of a finite set $V \subseteq \mathsf{Const} \cup \mathsf{LabNull} \cup \mathsf{UP}$ of *node identifiers* (or *nodes* for short) and a finite set of labeled edges (or *facts*) $E \subseteq V \times \Sigma \times V$, such that:

- if $(v_1, p, v_2) \in E$ and $p \in \mathsf{BP} \setminus \{\mathsf{type}\}$, then $v_1, v_2 \in \mathsf{Const} \cup \mathsf{LabNull}$,
- if $(v_1, \mathsf{type}, v_2) \in E$ then $v_1 \in \mathsf{Const} \cup \mathsf{LabNull}$, and $v_2 \in \mathsf{UP}$.

In first order logic (FOL), we use the following notation for edges:

- for (v, type, C), where $C \in \mathsf{UP}$, we use $C(v)$,
- for $(v_1, \approx, v_2)$ we use $v_1 \approx v_2$,
- for $(v_1, P, v_2$, where $P \in \mathsf{BP}$, we use $P(v_1, v_2)$.

A *rule* is a FOL sentence (implication) of the form $\forall \mathbf{x} \forall \mathbf{y} (\varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \psi(\mathbf{x}, \mathbf{z}))$, where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are tuples of variables. Formulas φ (the *body*) and ψ (the *head*) are conjunctions of atoms of the form $C(v)$, $P(v_1, v_2)$, and $v_1 \approx v_2$, where v, v_1, v_2 ranges over $\mathsf{Const} \cup \mathsf{LabNull}$. If the tuple $\mathbf{z}$ of existentially quantified variables is empty, the rule is *Datalog rule*. By $G \cup R$ we denote all facts belonging to G and deduced from G using rules in R.

An *ontology* (or a *knowledge base*) is a triple $O = (\Sigma, R, G)$, where Σ is a signature, R is a finite set of rules, and G is a database graph (a set of facts). A kind of ontology depends on the form of rules. For example, OWL 2 defines three profiles with different computational properties (OWL 2 Web Ontology Language Profiles, 2009).

A *query* is a FOL open formula. If the formula is constructed only with: (a) atoms of the form $C(v)$, $P(v_1, v_2)$ and $v \approx a$, where v, v_1, v_2 are variables, and a is a constant or labeled null; (b) symbols of conjunction ($\wedge$), disjunction ($\vee$), and existential quantification ($\exists$), then the query is a *positive existential query* (PEQ). A PEQ is *monadic* if has exactly one free variable, and is *conjunctive query* (CQ) if disjunction does not occur in this query.

A query $Q(\mathbf{x})$, where $\mathbf{x}$ is a tuple of free variables, is *satisfiable* in $O = (\Sigma, R, G)$, denoted $O \models Q(\mathbf{x})$ if Q is built from predicates in Σ, and there is a tuple $\mathbf{a}$ of elements from $\mathsf{Const} \cup \mathsf{LabNull}$ such that $G \cup R \models Q(\mathbf{a})$. Then $\mathbf{a}$ is an *answer* to $Q(\mathbf{x})$ with respect to O. Set of answers will be denoted by $Ans(Q)$.

3 FACETED QUERIES

There is an increasing number of data centered systems based on RDF and OWL 2. A standard query languages in such systems is SPARQL. This language, however, is not a convenient to end-users. As a more suitable interface for end-user data access have been developed approaches based on so-called *faceted search*. Now, we will define *faceted queries* for search over RDF graphs, and we will consider answering such queries when a database is additionally enhanced with an ontology. The considered system is a data integration system, where the graph data must be composed from data graphs stored in local databases.

In (Arenas et al., 2014), a *facet* is defined as a pair: $F = (X, \wedge\Gamma)$ (*conjunctive* facet), or $F = (X, \vee\Gamma)$ (*disjunctive* facet), where:

- $X \in \mathsf{BP}$ is the *facet name*, denoted by $F|_1$,
- Γ defines a set of *facet values* and is denoted $F|_2$,
- if $X = \mathsf{type}$, then $\Gamma \subseteq \mathsf{UP}$,
- if $X \in \mathsf{BP} \setminus \{\mathsf{type}\}$, then $\Gamma \subseteq \mathsf{Const} \cup \{\mathsf{any}\}$ or $\Gamma \subseteq \mathsf{UP} \cup \{\mathsf{any}\}$.

Any faceted query can be represented by a user-friendly graphical interface. A graphical form of faceted query in Figure 1, searches for ACM authors from NY university who have written a publication in year 2014.

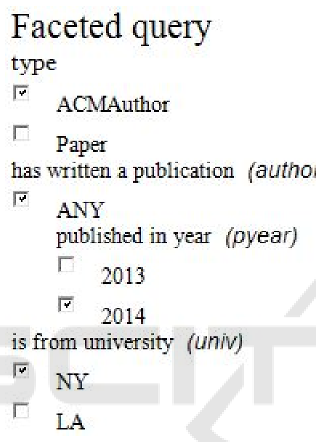
Faceted query
type
☑ ACMAuthor
☐ Paper
has written a publication *(authorOf)*
☑ ANY
published in year *(pyear)*
☐ 2013
☑ 2014
is from university *(univ)*
☑ NY
☐ LA

Figure 1: A graphical form of a faceted query.

Example 3.1. *For the considered example, the following facets can be defined:*

$$\begin{aligned}
F_1 &= (\mathsf{type}, \vee\{ACMAuthor, Paper\}),\\
F_2 &= (authorOf, \vee\{\mathsf{any}, p_1, a_1, a_2\}),\\
F_3 &= (pyear, \vee\{\mathsf{any}, 2013, 2014\}),\\
F_4 &= (univ, \vee\{\mathsf{any}, NY, LA\}),\\
F_5 &= (uinv, \wedge\{NY, LA\}).
\end{aligned}$$

Note, that F_5 is a conjunctive facet and denotes individuals which are simultaneously from two universities – NY and LA.

Definition 3.2. *Let $F = (X, \circ\Gamma)$, $\circ \in \{\wedge, \vee\}$, be a facet. A basic faceted query determined by F is a pair of the form $Q = (X, S)$, where $S \subseteq \Gamma$. A basic faceted query will be denoted by Q_t, if $X = \mathsf{type}$, and by Q_b when $X \in \mathsf{BP} \setminus \{\mathsf{type}\}$. A faceted query (or query for short) is an expression Q conforming to the following grammar:*

$$\begin{aligned}
Q &::= q \mid (q \wedge q) \mid (q \vee q)\\
q &::= Q_t \mid Q_b \mid (Q_b/Q)
\end{aligned}$$

Example 3.3. *The faceted query corresponding to this in Figure 1 is:*

$$\begin{aligned}
Q = (&(F_1, \{ACMAuthor\}) \wedge (F_4, \{NY\}))\\
&\wedge((F_2, \{\mathsf{any}\})/(F_3, \{2014\}))
\end{aligned} \qquad (1)$$

In Definition 3.4, we define semantics for faceted queries. The semantics $[\![Q(x)]\!]$ assigns to each query Q and a given variable x, a monadic PEQ with one free variable x.

Definition 3.4. *Let Q_t be a basic faceted query over $F_t = (\mathsf{type}, \circ\Gamma)$, Q_b be a basic faceted query over $F_P = (P, \circ\Gamma)$, $P \in \mathsf{BP} \setminus \{\mathsf{type}\}$, and Q be an arbitrary faceted query. Then semantics of faceted queries is defined as follows:*

1. *$Q_t = (F_t, S)$, $S \subseteq \mathsf{UP}$:*
$$[\![Q_t(x)]\!] = \underset{C \in S}{\circ} C(x).$$
2. *$Q_b = (F_P, \{\mathsf{any}\})$:*
$$[\![Q_b(x)]\!] = \exists y\, P(x, y),$$
$$[\![(Q_b/Q)(x)]\!] = \exists y\, P(x, y) \wedge [\![Q(y)]\!].$$
3. *$Q_b = (F_P, S)$, $S \subseteq \mathsf{Const} \cup \mathsf{LabNull}$:*
$$[\![Q_b(x)]\!] = \underset{a_i \in S}{\circ} \exists y_i\, P(x, y_i) \wedge y_i \approx a_i,$$
$$[\![(Q_b/Q)(x)]\!] = \underset{a_i \in S}{\circ} \exists y_i\, P(x, y_i) \wedge y_i \approx a_i \wedge [\![Q(y_i)]\!].$$
4. *$Q_b = (F_P, S)$, $S \subseteq \mathsf{UP}$:*
$$[\![Q_b(x)]\!] = \underset{C_i \in S}{\circ} \exists y_i\, P(x, y_i) \wedge C_i(y_i),$$
$$[\![(Q_b/Q)(x)]\!] = \underset{C_i \in S}{\circ} \exists y_i\, P(x, y_i) \wedge C_i(y_i) \wedge [\![Q(y_i)]\!].$$
5. *$Q_b = (F_P, \{\mathsf{any}\} \cup S)$, :*
$$[\![Q_b(x)]\!] = [\![(F_P, \{\mathsf{any}\})(x)]\!] \circ [\![(F_P, S)(x)]\!],$$
$$\begin{aligned}[\![(Q_b/Q)(x)]\!] = \;& [\![((F_P, \{\mathsf{any}\})/Q)(x)]\!]\\ &\circ [\![((F_P, S)/Q)(x)]\!].\end{aligned}$$
6. *Let q_1 and q_2 be queries, then:*
$$[\![(q_1 \wedge q_2)(x)]\!] = [\![q_1(x)]\!] \wedge [\![q_2(x)]\!],$$
$$[\![(q_1 \vee q_2)(x)]\!] = [\![q_1(x)]\!] \vee [\![q_2(x)]\!].$$

□

The semantics of basic type-faceted queries of the form (F, S) is the conjunction (disjunction) of atoms of the form $C(x)$ over the same variable, where $C \in S$. If the facet name is a binary predicate P, then the query is translated to: (a) an atom whose second argument is existentially quantified (if any occurs); (b) a conjunction (disjunction) of binary atoms whose second argument must be equal to a constant or a labeled null, or must satisfy an unary predicate. In the case of nesting, a variable from the parent is shifted to the child. Finally, conjunction (disjunction) of queries is interpreted as the conjunction (disjunction) of the corresponding formulas.

The first order interpretation (the PEQ) of faceted query (1) is given in (2).

$$\begin{aligned}[\![Q(x)]\!] = \quad & ACMAuthor(x) \\ & \wedge \exists y(authorOf(x,y) \\ & \quad \wedge \exists z(pyear(y,z) \wedge z \approx 2014)) \\ & \wedge \exists w(univ(x,w) \wedge w \approx NY).\end{aligned} \tag{2}$$

4 ONTOLOGY-BASED DATA INTEGRATION

Ontology Based Data Integration (OBDI), or Ontology Based Data Access (OBDA) involves the use of ontology to effectively combine data or information from multiple heterogeneous sources (Wache et al., 2001). In this paper we will follow so called *single ontology approach*, i.e., an approach when a single ontology is used as a global reference model in the system. We assume that the data integration system is based on a global schema (Ullman, 1997), (Halevy et al., 2006), (Lenzerini, 2002) (another approaches assume P2P data integration, see for example (Calvanese et al., 2004)).

On the conceptual level, a user perceives contents of the system as a large single ontology $O = (\Sigma, R, G)$, and formulates faceted queries against this ontology. On the implementation level, we assume that (see Figure 2):

1. The (global) schema of the system consists of the signature and the set of rules of the ontology, i.e., $Sch = (\Sigma, R)$.
2. Facts, represented by means of RDF graphs, are stored in local databases, $DB_i = (\Sigma_i, G_i)$, where $\Sigma_i \subseteq \Sigma$ consists of symbols, i.e., unary and binary predicates, occurring in RDF graph G_i.
3. Data in different local databases complement each other and can overlap. We assume, however, that databases are consistent and do not contradict one another.
4. Local databases are created by local users and the global schema is used as the reference in introducing new facts.

A user query is rewritten and sent to each database in a form understandable and executable to this database management system. Next, partial answers are sent back, merged accordingly and finally returned to the user.Answering queries requires some data inferring processes implied by ontology deductive rules. The architecture of such a system is given in Figure 2.

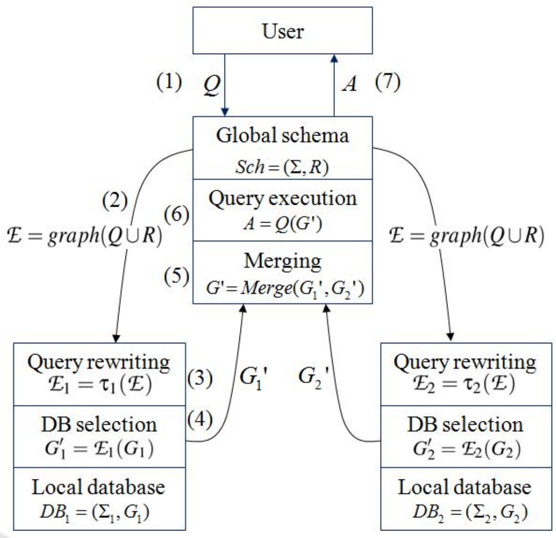

Figure 2: Architecture of an ontology-based data integration.

The system works as follows:

(1) The user formulates a faceted query Q.

(2) Q is translated to a FOL formula $[\![(Qx)]\!]$ and its graph representation is created. This graph is extended to $\mathcal{E}$ with elements corresponding to relevant deductive rules in the schema. Graph $\mathcal{E}$ is sent to local database management systems.

(3) $\mathcal{E}$ is reduced to $\mathcal{E}_i$ using information from signature Σ_i.

(4) A set of edges is selected from G_i, which are relevant to query answering.

(5) Selected subgraphs are merged into graph G'.

(6) The user query Q is evaluated over G', and the answer A is obtained.

(7) A is returned to the user.

Example 4.1. *The global schema, $O = (\Sigma, R)$, relevant to our example can contain the following set of deductive rules:*

$$\begin{array}{ll}(R1) & atConf(x,y) \wedge y \approx ACMConf \\ & \qquad \rightarrow ACMPaper(x), \\ (R2) & authorOf(x,y) \wedge ACMPaper(y) \\ & \qquad \rightarrow ACMAuthor(x), \\ (R3) & atConf(x,y) \wedge cyear(y,z) \rightarrow pyear(x,z), \\ (R4) & univ(x,y_1) \wedge univ(x,y_2) \rightarrow y_1 \approx y_2, \\ (R5) & title(x_1,y) \wedge title(x_2,y) \rightarrow x_1 \approx x_2, \\ (R6) & Author(x) \rightarrow \exists y(authorOf(x,y) \wedge Paper(y)).\end{array}$$

Sample RDF graphs, G_1 and G_2, of two local databases are given in, respectively, Figure 3 and Figure 4. These graphs are built over signatures, respectively, Σ_1 and Σ_2, being subsets of Σ, and over a set of constants, Const, and a set of labeled nulls LabNull.

In this case $John, Ann, 2014, 2013, KB, AI, NY, LA$, $ACMConf$, and $IEEEConf$ are in Const, and p_1, a_1, a_2 are in LabNull. Labeled nulls are used as identifiers of anonymous nodes and can be replaced with other labeled nulls or with constants. So, they are like variables (Fagin et al., 2005).

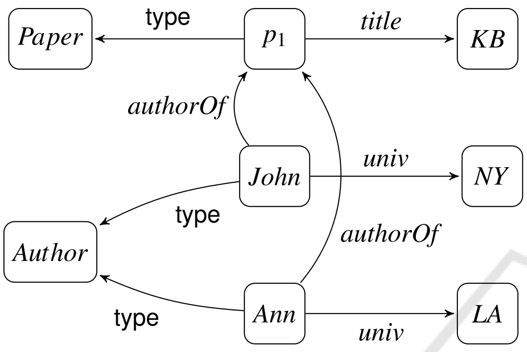

Figure 3: RDF graph G_1 of a local database DB_1.

The FOL form of G_1:

$Paper(p_1), title(p_1, KB), Author(John),$
$Author(Ann), authorOf(John, p_1),$
$authorOf(Ann, p_1), univ(John, NY), univ(Ann, LA),$

and of G_2:

$Paper(a_1), Paper(a_2), title(a_1, KB),$
$atConf(a_1, ACMConf), cyear(ACMConf, 2014),$
$title(a_2, AI), atConf(a_2, IEEEConf),$
$cyear(IEEEConf, 2013), Author(Ann),$
$authorOf(Ann, a_1), authorOf(Ann, a_2).$

If we evaluate query (2) against G_1 or/and G_2, then the answer is empty (in particular, the binary relation $ACMAuthor$ does not even exist). However, if we consider also the set of rules in the ontology and apply them to infer new facts, we see that (2) is satisfied by $John$. So $John$ is the answer to the query under consideration. Thus, to obtain the answer we have to:

- merge database states,
- take into account deductive rules from the ontology,
- apply deductive rules to infer new facts from the result of merging,
- evaluate the query over the set of all facts.

Note, however, that a naive performance of these operations can be rather inefficient. For example, we

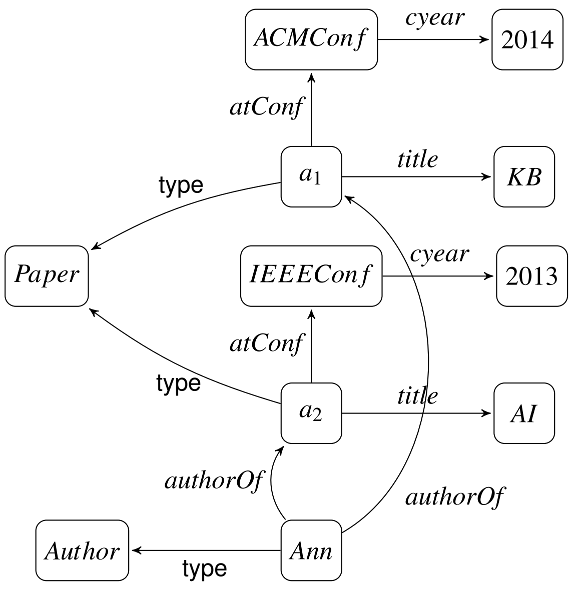

Figure 4: RDF graph G_2 of a local database DB_2.

can merge whole database states – which is rather very inefficient, or we can take into consideration only such subgraphes which are relevant to obtain the answer. Further on in the paper, we will discuss how these relevant subgraphs can be chosen.

5 ANSWERING FACETED QUERIES

5.1 Creating Global Query Patterns

Now, we will discuss the problem of selecting some facts (edges) from RDF graphs (step (4) in Figure 2) which should be sent to the merge stage (step (5) in Figure 2). The general assumption about the selection is that there must be *justification* to select an edge. The selection of an edge (x, P, y) is justified if:

- predicate P occurs in the query;
- predicate P occurs in the left hand site of an ontology rule, and there is a justification to select a predicate P' occurring on the right hand site of this rule;
- P is functional or a key (P^- is functional) and can be used to infer an equality between some data involved in the answer to the query.

A facet graph $\mathcal{G}$ for a facet query Q represented by a FOL formula $[\![Q(x)]\!]$, is the graph $\mathcal{G} = (V, E)$, where:

1. V is a set of unary predicate names, variable names, constants and labeled nulls, occurring in

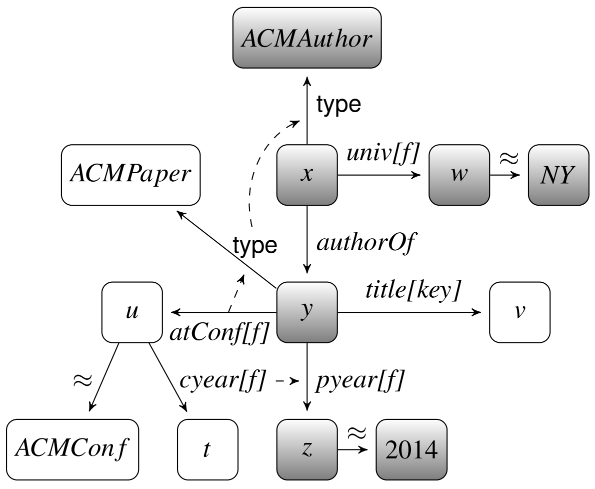

Figure 5: Extended query graph $\mathcal{E}$.

$[\![Q(x)]\!]$.

2. The set E of edges is defined as follows:
   - if $C(v)$ is in $[\![Q(x)]\!]$, then (v, type, C) is in E,
   - if $P(v_1, v_2)$ is in $[\![Q(x)]\!]$, then (v_1, P, v_2) is in E,
   - if $v \approx a$ is in $[\![Q(x)]\!]$, then $(v, \approx, a)$ is in E.

In Figure 5, the subgraph with greyed nodes constitutes the query graph for the faceted query in Figure 1 and its FOL interpretation (2). Additionally, in Figure 5 some edges are qualified with: $[f]$, to denote that the corresponding binary predicate is a *function* (rule $(R4)$), and $[key]$, to denote that the corresponding binary predicate is a *key*, i.e., its inversion is a function (rule $(R5)$).

Next, the query graph $\mathcal{G}$ is extended to an *extended query graph* (or *global query pattern*), $\mathcal{E} = (V_{\mathcal{E}}, E_{\mathcal{E}})$, by adding some edges implied by ontology rules. We take into account rules which are adjacent to the current form of the extended query graph. We proceed as follows:

1. We start with assuming $\mathcal{E} = (V_{\mathcal{E}}, E_{\mathcal{E}})$ equal to $\mathcal{G} = (V, E)$.
2. Let $\varphi \to C(v)$ be a rule and (x, type, C) be in $E_{\mathcal{E}}$, for some variable x. Then:
   - rename all variables occurring in φ, with the exception of variable v, in such a way that new names are different from those occurring in $V_{\mathcal{E}}$;
   - rename v to x,
   - match the renamed form of φ to edges in $E_{\mathcal{E}}$, and rename variables accordingly. The result denote by φ',
   - extend $E_{\mathcal{E}}$ as follows:
     – if $C(w)$ is in φ' and not in $E_{\mathcal{E}}$, then add the edge (w, type, C) to $E_{\mathcal{E}}$,
     – if $P(w_1, w_2)$ is in φ' and not in $E_{\mathcal{E}}$, then add (w_1, P, w_2) to $E_{\mathcal{E}}$,
     – if $w \approx a$ is in φ' and not in $E_{\mathcal{E}}$, add $(w, \approx, a)$ to $E_{\mathcal{E}}$.
3. Let $\varphi \to P(v_1, v_2)$ be a rule and (x, P, y) be in $E_{\mathcal{E}}$, for some variables x and y. Then:
   - rename all variables occurring in φ, with the exception of variables v_1 and v_2, in such a way that new names are different from those occurring in $V_{\mathcal{E}}$;
   - rename v_1 to x, and v_2 to y,
   - match the renamed form of φ to edges in $E_{\mathcal{E}}$, and rename variables accordingly. The result denote by φ',
   - extend $E_{\mathcal{E}}$ analogously to the extension described in (2).
4. Let φ be a rule defining functionality of a binary predicate $P(v_1, v_2)$. Let x be a variable in $V_{\mathcal{E}}$ defined over the range of P. Then rename v_2 to x, and v_1 to an appropriate name w, and add (w, P, x) to $E_{\mathcal{E}}$.
5. Let φ be a rule defining functionality of inversion of a binary predicate $P(v_1, v_2)$, i.e., determining that P is a key. Let x be a variable in $V_{\mathcal{E}}$ defined over the domain of P. Then rename v_1 to x, and v_2 to an appropriate name w, and add (x, P, w) to $E_{\mathcal{E}}$.

In Figure 5, edges with white nodes were added according to the above procedure. Dashed arrows indicate which edges are needed to infer another edges. In particular, $(y, \mathsf{type}, ACMPaper)$ is necessary to infer $(x, \mathsf{type}, ACMAuthor)$ (rule $(R2)$). To infer $(y, \mathsf{type}, ACMPaper)$, we need $(y, atConf, u)$ and $(u, \approx, ACMConf)$ (rule $(R1)$). To infer $(y, pyear, z)$, the edge $(u, cyear, t)$ is needed, (rule $(R3)$). Finally, $(y, title, v)$ is added since $title$ is a *key*, i.e., its inversion, $title^{-}$, is a function (rule $(R5)$).

5.2 Local Answers to Graph Patterns

Restrictions of graph $\mathcal{E}$ (Figure 5) to DB_1 and DB_2 are extended graphs (*local query patterns*), $\mathcal{E}_1$ and $\mathcal{E}_2$, presented in Figure 6 and Figure 7, respectively.

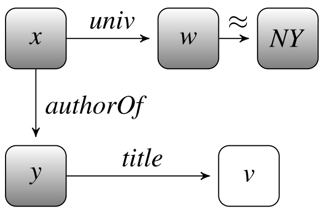

Figure 6: Extended query graph $\mathcal{E}_1 = \tau_{\Sigma_1}(\mathcal{E})$.

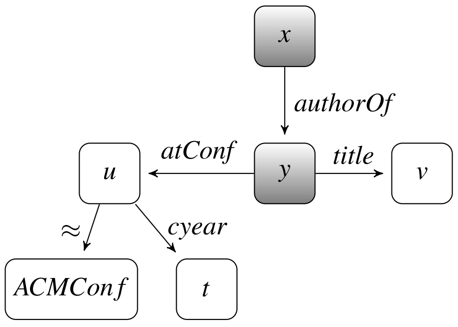

Figure 7: Extended query graph $\mathcal{E}_2 = \tau_{\Sigma_2}(\mathcal{E})$.

Subgraphs $G'_1 = \mathcal{E}_1(G_1)$ and $G'_2 = \mathcal{E}_2(G_2)$, which are answers to pattern queries $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively, are presented in Figure 8 and Figure 9, respectively.

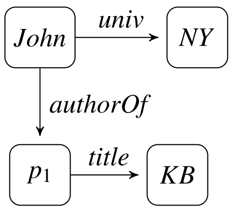

Figure 8: Answer $G'_1 = \mathcal{E}_1(G_1)$.

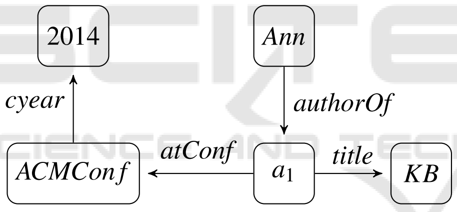

Figure 9: Answer $G'_2 = \mathcal{E}_2(G_2)$.

Next, RDF subgraphs G'_1 and G'_2, are sent to the merging service.

5.3 Merging Local Answers

Partial answers, like G'_1 and G'_2, must be merged to produce a RDF graph over which the user query Q can be evaluated. Now, we propose a method to perform the merging. The merge is done by means of mapping rules produced from the extended query graph $\mathcal{E}$ and from the set R of ontology rules belonging to the global schema. These rules are used to define the *chase procedure* as it was proposed in data exchange theory (Fagin et al., 2005), (Calvanese et al., 2007b). Predicates prefixed by s refer to source data, i.e., to G'_1 and G'_2. Predicates without prefixes, refer to target data, i.e., to the result of the merge, and are understood as targed constraints. In our case, the set of generated mapping rules used for merging is given in Figure 10.

$$\begin{array}{l}
s.authorOf(x,y) \rightarrow authorOf(x,y),\\
s.univ(x,y) \rightarrow univ(x,y),\\
s.title(x,y) \rightarrow title(x,y),\\
s.atConf(x,y) \wedge y \approx ACMConf \rightarrow ACMPaper(x),\\
authorOf(x,y) \wedge ACMPaper(y) \rightarrow ACMAuthor(x),\\
s.atConf(x,y) \wedge s.cyear(y,z) \rightarrow pyear(x,z),\\
title(x_1,y) \wedge title(x_2,y) \rightarrow x_1 \approx x_2.
\end{array}$$

Figure 10: Mapping rules used in merging.

In particular, the last rule enforces $a_1 \approx p_1$. So, in the result RDF graph all occurrences of a_1 are replaced by p_1. The result of merge is given in Figure 11.

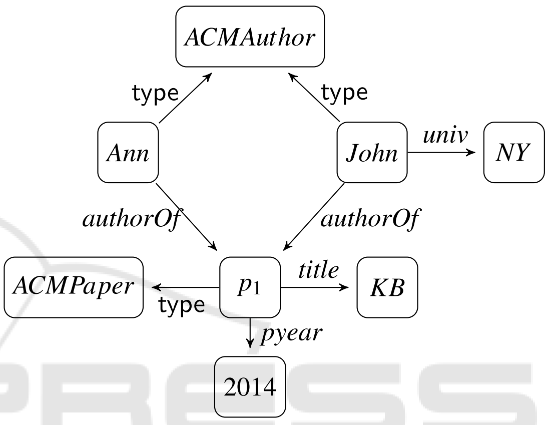

Figure 11: Result of merging, $G' = Merge(G'_1, G'_2)$, by means of mapping rules from Figure 10.

5.4 Obtaining Final Answers

The result of merging of local answers to local graph patterns, as $G' = Merge(G'_1, G'_2)$ in Figure 11, constitutes a dataset which is the object to evaluate a faceted query under consideration. The first order representation of the query, in our case (2), is a monadic PEQ resulting from a faceted query. So, the answer can be found in polynomial time. It is easily seen that the answer is *John*.

So called *refocussing* functionality in faceted queries allows for changing the free variable of the query Q. In consequence, the answer consists of all valuations of this free variable. In our example, if we want to now information about papers written by ACM authors, we should refocus our attention to the variable being the second argument of $authorOf$ predicate.

6 CONCLUSION

In this paper, we have discussed an ontology-based data integration system with faceted query interface. In such a system we have both, *extensional* and *intentional* knowledge. The extensional knowledge is stored as RDF graphs in local databases, and the intentional knowledge is given as a set of rules constituting a set of axioms of a global ontology. A user formulates faceted queries in a user-friendly way using a simple graphical interface. Next, local databases are queried about data which is indirectly or directly (to infer new facts by means of ontology rules) necessary to answer the query. The set of local answers are merged and finally the expected answer is obtained. The proposed method is a base to introduce new functionality into our system of data integration.

REFERENCES

Arenas, M., Grau, B. C., Kharlamov, E., Marciuska, S., and Zheleznyakov, D. (2014). Faceted search over ontology-enhanced RDF data. In *ACM CIKM 2014*, pages 939–948. ACM.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Petel-Schneider, P., editors (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Barceló, P. and Fontaine, G. (2015). On the data complexity of consistent query answering over graph databases. In *ICDT 2015*, volume 31 of *LIPIcs*, pages 380–397. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Bernstein, P. A. and Haas, L. M. (2008). Information integration in the enterprise. *Commun. ACM*, 51(9):72–79.

Calì, A., Calvanese, D., Giacomo, G. D., and Lenzerini, M. (2004). Data integration under integrity constraints. *Information Systystems*, 29(2):147–163.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., and Rosati, R. (2007a). Ontology-based database access. In *SEBD 2007*, pages 324–331.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R., and Ruzzi, M. (2010). Using OWL in Data Integration. In *Semantic Web Information Management. Chapter 17*, pages 397–424. Springer.

Calvanese, D., Giacomo, G. D., et al., (2007b). EQL-Lite: Effective First-Order Query Processing in Description Logics. In *IJCAI, International Joint Conference on Artificial Intelligence*, pages 274–279.

Calvanese, D., Giacomo, G. D., Lenzerini, M., and Rosati, R. (2004). Logical Foundations of Peer-To-Peer Data Integration. In *PODS*, pages 241–251.

Cruz, I. F. and Xiao, H. (2009). Ontology driven data integration in heterogeneous networks. In *Complex Systems in Knowledge-based Environments*, pages 75–98.

Das, S., Chong, E., Eadon, G., and Srinivasan, J. (2004). Supporting Ontology-Based Semantic Matching in RDBMS. In *Proc. of the 30th International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada*, pages 1054–1065.

Eklund, P. W., II, R. J. C., and Roberts, N. (2004). Retrieving and exploring ontology-based information. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 405–414. Springer.

Fagin, R., Haas, L. M., Hernández, M. A., Miller, R. J., Popa, L., and Velegrakis, Y. (2009). Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*, volume LNCS **5600**, pages 198–236.

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comput. Sci*, 336(1):89–124.

Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H., and Scheel, U. (2010). Faceted Wikipedia Search. In *BIS 2010*, volume 47 of *Lecture Notes in Business Information Processing*, pages 1–11. Springer.

Halevy, A. Y., Rajaraman, A., and Ordille, J. J. (2006). Data integration: The teenage years. In Dayal, U., Whang, K.-Y., Lomet, D. B., Alonso, G., Lohman, G. M., Kersten, M. L., Cha, S. K., and Kim, Y.-K., editors, *VLDB*, pages 9–16. ACM.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In Popa, L., editor, *PODS*, pages 233–246. ACM.

Oren, E., Delbru, R., and Decker, S. (2006). Extending faceted navigation for RDF data. In *ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 559–572. Springer.

OWL 2 Web Ontology Language Profiles (2009). www.w3.org/TR/owl2-profiles.

Resource Description Framework (RDF) Model and Syntax Specification (1999). www.w3.org/TR/PR-rdf-syntax/.

Skjæveland, M. G., Giese, M., Hovland, D., Lian, E. H., and Waaler, A. (2015). Engineering ontology-based access to real-world data sources. *J. Web Sem.*, 33:112–140.

SPARQL Query Language for RDF (2008). http://www.w3.org/TR/rdf-sparql-query.

Ullman, J. D. (1997). Information integration using logical views. in: Database Theory - ICDT 1997. *Lecture Notes in Computer Science*, 1186:19–40.

Wache, H., Vgele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hbner, S. (2001). Ontology-Based Integration of Information - A Survey of Existing Approaches. In *IJCAI 2001*, pages 108–117.

Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. (2003). Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 401–408. ACM.