Exploring Ontology-Enhanced Bibliography Databases Using Faceted Search

Tadeusz Pankowski(✉)

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

Abstract. In this paper, we exploited the application of the Ontology Based Data Access (OBDA) approach, equipped with faceted search utilities, to explore bibliography databases. A bibliography database is enhanced by means of an ontology leading to a bibliography information space. We show that faceted search paradigm to explore such information space is particularly attractive. We describe an implementation of this approach in DAFO system. We focus on formulating faceted queries over the ontology, mapping the ontology to a relational database, and on transforming the query to executable forms. The final version of a faceted query is a SQL query that is executed in a relational database system. The computational results show that the usage of faceted search-oriented way of modeling and retrieving information is very promising.

1 Introduction

Faceted search is commonly used in retrieving data in e-commerce applications [7,17]. In this paper, we adapt this approach to explore bibliography databases. To take advantages of this retrieval paradigm, a bibliography database should be first enriched with an ontology, leading to the creation of an ontology-enhanced bibliography database. The purpose of this extension is to provide the database with concepts, relationships and rules, which both facilitate query formulation and allow for a flexible perceiving the domain information space. The main advantages of the faceted search are: (a) iterative and interactive support for query formulation, usually based on a user-friendly graphical interface; (b) coexistence of many different views over the underlying information space; (c) effective implementation due to the expression power of faceted queries limited to first order monadic positive existential queries (MPEQ) [1,11,16].

Related work. This paper refers both to Ontology-Based Data Access (OBDA) and faceted search. In OBDA, an ontology is used as a global schema and a database is used as a data repository and a mapping is established between the ontology and the database [3,13]. Some issues concerning this approach were discussed in data integration and data exchange contexts [5,8]. However, the problem in such system is a query language. It is unrealistic to require the user to know the database schema in details, so the usage of SQL, SPARQL or XQuery as

© Springer International Publishing AG 2017
J. Kamps et al. (Eds.): TPDL 2017, LNCS 10450, pp. 27–39, 2017.
DOI: 10.1007/978-3-319-67008-9_3

end-user languages is unacceptable. On the other hand, relying only on keyword queries (even with boolean operators) significantly limits search capabilities. Thus, it is quite obvious that users should be provided with graphical-oriented tools. Such solution can be based on Query-By-Example [9,18], which was the inspiration for developing a number of visual query systems and languages [4]. Visual systems provide an intuitive and natural perceiving of the information space, and follow the direct manipulation idea with visual representation of domain and query manipulation. End users recognize the relevant fragments of information space and formulate queries by directly manipulating them. To this family of information retrieval paradigms we can count faceted search. The faceted search combines two classical approaches, namely keyword-based search and manipulation search with narrowing the information space.

Contribution. We discuss the aforementioned issues in the context of our system called DAFO (Data Access with Faceted queries over Ontology) [14,15]. The main contributions of this paper are as follows: (1) we discuss an ontology based on OWL 2 RL to describe an information space relevant to bibliography database; (2) we show how the ontology is used in faceted query formulation; (3) we describe main steps in answering faceted queries: (a) translating to first order faceted queries (FOFQ), (b) rewriting faceted queries using ontology rules, (c) mapping the ontology into relational database; (4) we report some computational experiments which prove that the formulation and evaluation of faceted queries in DAFO is very promising.

Paper outline: The structure of the paper is the following. Ontology-enhanced databases are discussed in Sect. 2. In Sect. 3, a mapping of the ontology into relational database is defined. Faceted search over bibliography ontology and some experimental results are presented in Sect. 4. Section 5 summarizes the paper.

2 Ontology-Enhanced Bibliography Database

2.1 Relational Schema of Bibliography Database

In this section we motivate our research showing the advantages of combining information representation capabilities provided by relational database and an ontology. The discussion will be focused on a bibliography database, BibDb, with the schema in Fig. 1.

The schema was designed based on analysis of DBLP Computer Science Bibliography [6,10]. An instance of the database was prepared by extracting data from DBLP resources (from XML, HTML, and BibTex files), and enriched with data extracted form personal and conference home pages. Some tables have primary keys (denoted by Id) used to identify entities and to establish cross references between tables. Same tables have also unique DBLP identifiers ($DblpKey$) used as references to DBLP bibliography.

The schema in Fig. 1 can be used as a target schema for posting relational queries. However, direct operating on such schema is troublesome. In general, the schema can be large, incomprehensible, and a language for query formulation

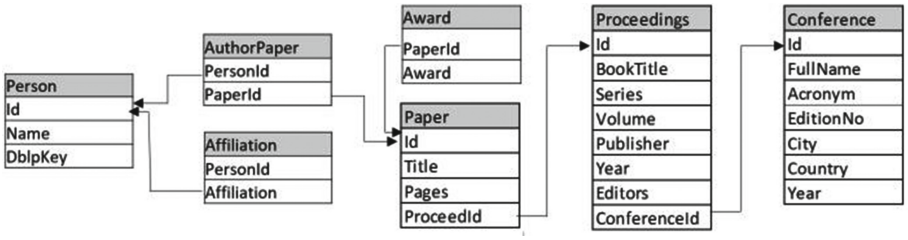

Fig. 1. Diagram of bibliography relational database BibDb.

can make requirements that are difficult to accept. For example, it is unrealistic to demand that the user can write SQL queries.

Formally, a relational database schema is a tuple $\mathcal{S} = (\mathbf{R}, att, pkey, InclDep)$, where $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a set of relation names; att assigns a set of attributes to each $R \in \mathbf{R}$, $att(R) \subseteq \mathsf{Att}$; $pkey$ assigns a primary key to some $R \in \mathbf{R}$, $pkey(R) \in att(R)$; $InclDep$ is a set of *inclusion dependencies* (or *referential constraints*), i.e., expressions of the form $R[A] \subseteq R'[A']$, where $A \in att(R)$, $A' = pkey(R')$, and A is called a *foreign key* referring from R to R'.

In Fig. 1 there is a graphical representation of relational database schema. By Id we denote primary keys, and inclusion dependencies are denoted by arrows.

2.2 Ontology Describing Bibliography Information Space

By a *bibliography information space* we understand a specification of the knowledge relevant to the bibliography domain. In practice, this knowledge covers and enriches relational schema, and is defined by means of an *ontology*. Then we say about *ontology-enhanced* database.

An ontology [2,11] is a pair $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ and $\mathcal{A}$ are, respectively, the *terminological* and *assertional* parts of the ontology. The terminological part is a pair $\mathcal{T} = (\Sigma, \mathcal{R})$, where $\Sigma = \mathsf{UP} \cup \mathsf{BP} \cup \mathsf{Const}$ is the *signature* of the ontology, and specifies a set of *unary predicates* (UP), a set of *binary predicates* (BP), and a set of constants (Const). $\mathcal{R}$ is a set of ontology *rules*. The assertional part is a set of *assertions* (facts), i.e., expressions of the form $C(a)$ or $P(a, b)$, where $C \in \mathsf{UP}$, $P \in \mathsf{BP}$, and $a, b \in \mathsf{Const}$. The set UP of unary predicates is divided into *extensional* (UP_E) and *intentional* (UP_I) ones. Similarly, BP is divided into BP_E and BP_I. Extensional predicates are those, which appear in $\mathcal{A}$, while intentional predicates do not appear explicitly in $\mathcal{A}$ but are defined by means of rules in $\mathcal{R}$.

For example, the considered bibliography information space can be defined by means of on ontology BibOn. A fragment of terminological part of BibOn is depicted in Fig. 2. By solid lines we drawn extensional predicates, and intentional predicates are denoted by dashed lines.

Rules in ontologies usually conform to those specified in OWL 2 profiles [12]. We will restrict ourselves to categories of rules given in Table 1. Additionally, following so called *extended knowledge bases* introduced in [11], we divide the

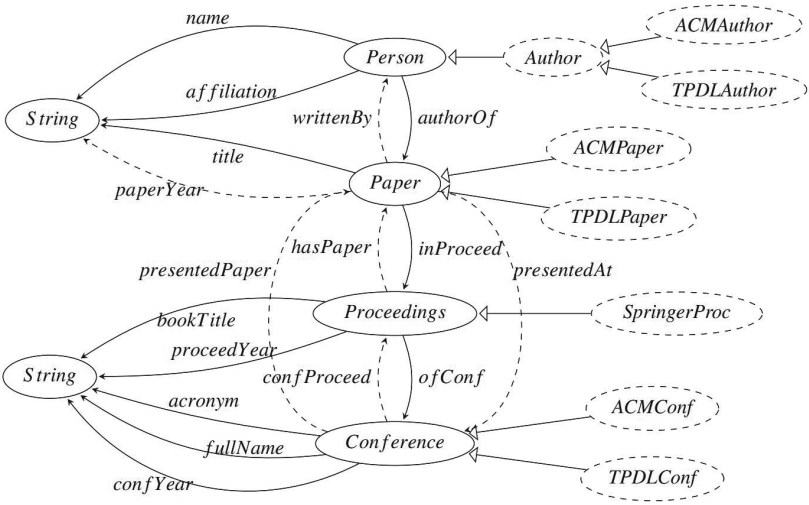

Fig. 2. Terminological part of BibOn ontology.

set of rules into *deductive rules* (Ded-1 – Ded-9) and *integrity constraints* (IC-1 – IC-4). A deductive rule is used to deduce (infer) new assertions (intentional or extensional) from extensional and already deduced intentional ones. Integrity constraints are used to check correctness of the given set of assertions, and not to infer new assertions. Note that all but the last two are in OWL 2 RL [12]. Moreover, functionality (IC-1) and key (IC-2) rules can be used as deductive rules in the case when labeled nulls are allowed (see [8,14,15]).

Deductive rules and integrity constraints referred to in this paper are:

1. *Deductive rules* – specify how some types or properties may be deduced from another. In this paper they are used to infer:
   (a) *inheritance hierarchies* (subtypes and subproperties), e.g., (a) a unary predicate (type) *Author* is a subtype of *Person*, and (b) binary predicate (property) *awardedAt* is a subproperty of *presentedAt*;
   (b) *domains* and *ranges* of binary predicates, e.g., property *authorOf* has *Person* as its domain (although may be not defined for all persons), and *Paper* as the range.
   (c) a *value-driven specialization* – a subtype C may be a subset of domain of P, for which P has a given value a, e.g., *SpringerProceed* is a subtype of *Proceedings*, for which *publisher* has value *Springer*;
   (d) a *pattern-driven specialization* – like value-driven specialization but now, a is treated as a pattern, and the value of P must conform to this pattern, e.g., *ACMConf* is a subtype of *Conference* if *acronym* of the conference contains "*ACM*", i.e., conforms to the pattern "*%ACM%*";
   (e) a *type-driven specialization* – a subtype C is a subset of domain of property P, for which P has value of a given type D, e.g., *ACMAuthor* is the type of those authors, who presented their papers at an *ACMConf*;

Table 1. Categories of ontology rules used in bibliography ontology BibOn.

Id	General form of a rule	Name	Representation
Ded-1	$\forall x\ (D(x) \rightarrow C(x))$	Subtype	$subtype(D, C)$
Ded-2	$\forall x, y\ (S(x, y) \rightarrow P(x, y))$	Subproperty	$subprop(S, P)$
Ded-3	$\forall x, y\ (P(x, y) \rightarrow C(x))$	Domain	$dom(P, C)$
Ded-4	$\forall x, y\ (P(x, y) \rightarrow D(y))$	Range	$rng(P, D)$
Ded-5	$\forall x\ (P(x, y) \wedge y = a \rightarrow C(x))$	Specialization (value and pattern driven)	$spec1(P, a, C)$
Ded-6	$\forall x, y\ (P(x, y) \wedge D(y) \rightarrow C(x))$	Specialization (type driven)	$spec2(P, D, C)$
Ded-7	$\forall x, y, z\ (R(x, y) \wedge S(x, z) \wedge z = a \rightarrow P(x, y))$	Property specialization (value and pattern driven)	$spec3(R, S, a, P)$
Ded-8	$\forall x, y, z\ (S(x, z) \wedge T(z, y) \rightarrow P(x, y))$	Chain (composition)	$chain(S, T, P)$
Ded-9	$\forall x, y\ (S(y, x) \rightarrow P(x, y))$	Inversion	$inv(S, P)$
IC-1	$\forall x, y_1, y_2\ (P(x, y_1) \wedge P(x, y_2) \rightarrow y_1 = y_2)$	Functionality	$func(P)$
IC-2	$\forall x_1, x_2, y\ (P(x_1, y) \wedge P(x_2, y) \rightarrow x_1 = x_2)$	Key (functionality of inversion)	$key(P)$
IC-3	$\forall x(C(x) \rightarrow \exists y\ P(x, y)$	Existence	$exists(C, P)$
IC-4	$\forall x(C(x) \rightarrow \exists y\ P(x, y) \wedge y = a$	Has value	$hasVal(C, P, a)$

(f) a *chain* (or *composition*) – a property is a chain (composition) of two other properties, e.g., *paperYear* (year of a paper) is the chain of *inProceed* and *proceedYear* (i.e., year of the proceedings the paper is in);
(g) an *inversion* – a property is the inversion of other property, e.g., *writtenBy* is the inversion of *authorOf*.

2. *Integrity constraints* – are used to check whether a given set of assertions is consistent:
   (a) *functionality* – states that a property P is a function, e.g., *name*, *inProceed*;
   (b) *key* – states that value of a property P uniquely identifies the domain object, i.e., inversion of P is a function, e.g., *title*, or *hasPaper* (paper uniquely identifies the proceeding in which the paper is included);
   (c) *existence* – states that a property P is defined on each element in C, e.g., $authorOf$ is defined on each object of *Author* type;
   (d) *has value* – states that a property P on each element in C has the same value a (or conforms to pattern a), e.g., the value of *acronym* for each object of *ACMConf* type conforms to the pattern "*%ACM%*".

A first order (FO) formula $\varphi(x)$ is a *monadic positive existential query* (MPEQ), if it has exactly one free variable and is constructed only of: (a) atoms of the form $C(x)$, $P(x_1, x_2)$ and $x = a$; (b) conjunction ($\wedge$), disjunction ($\vee$),

and existential quantification ($\exists$). A constant a is an *answer* to $\varphi(x)$ if $\varphi(a)$ is satisfied in $\mathcal{O}$.

Any rule can be treated either as deductive rule or as an integrity constraint. Our choice is motivated by the use of rules to rewrite queries. The following example explains the role of rules and integrity constraints in query rewriting.

Example 1. It seems obvious that a query $q_1(x) = Author(x)$ can be replaced with $q_2(x) = Person(x) \wedge \exists y(authorOf(x, y))$. We would expect that sets of answers to these queries were equal. However, this is the case only when the data source satisfies the integrity constraint (IC-3) $Author(x) \rightarrow \exists y(authorOf(x, y))$. Indeed, let
$\mathcal{A}=\{Person(a), Person(b), Person(c), Author(a), Author(c), authorOf(a, p)\}$.
Then $q_1(x)(\mathcal{A}) = \{a, c\}$, but $q_2(x)(\mathcal{A}) = \{a\}$. □

New rules can be dynamically added to the ontology. For example, we can add the binary predicate $authorConf$ connecting authors with conferences, which is the chain of $authorOf$, $inProceed$, and $ofConf$.

3 Mapping Ontology to Relational Database

In DAFO, queries formulated over an ontology are evaluated in a relational database. Thus, a mapping of the ontology into relational database must be defined. Two levels of the mapping are distinguished: (1) *metaschema level* – assigns schema elements to ontology predicates, and (2)*schema level* – assigns relational data to ontology assertions.

A mapping on a metaschema level is specified by four functions: *table*, *domCol*, *rngCol* and *inclDep*, defined as follows (some examples are given in Table 2).

1. Extensional unary predicates are mapped to relational names. Formally, if $C \in \mathsf{UP}_E$ then $table(C) = R_C \in \mathbf{R}$, and $domCol(C) = pkey(R_C)$; $R_C \neq R_{C'}$ for $C \neq C'$.
2. Extensional binary predicates are divided in four classes: *functional data* properties (BP_{fd}), *multivalued data* properties (BP_{md}), *functional object* properties (BP_{fo}), and *multivalued object* properties (BP_{mo}). Data properties are binary predicates with *String* as their ranges. Object properties have ranges different from *String*. Then
   - if $P \in \mathsf{BP}_{fd}$, and C is domain of P, then: $table(P) = R_C$, $domCol(P) = pkey(R_C) = Id$, $rngCol(P) = A^r_P \in att(R_C)$, e.g., *name*;
   - if $P \in \mathsf{BP}_{md}$, and C is domain of P, then: $table(P) = R_P \neq R_C$, $domCol(P) = A^d_P \in att(R_P)$, $rngCol(P) = A^r_P \in att(R_P)$, and $R_P.A^d_P \subseteq R_C.Id$, where A^d_P is a foreign key referring from R_P to R_C, e.g., *affiliation*;
   - if $P \in \mathsf{BP}_{fo}$, C is domain of P, D is range of P, then: $table(P) = R_C$, $domCol(P) = pkey(R_C) = Id$, $rngCol(P) = A^r_P \in att(R_C)$, and $R_C.A^r_P \subseteq R_D.Id$, where A^r_P is a foreign key referring from R_C to R_D, e.g., *inProceed*;

Table 2. Mapping predicates of ontology BibOn into database schema BibDb.

Name	Class	table	domCol	rngCol	inclDep
$Person$	UP_E	$Person$	Id		
$name$	BP_{fd}	$Person$	Id	$Name$	
$affiliation$	BP_{md}	$Affiliation$	$PersonId$	$Affiliation$	$Affiliation[PersonId] \subseteq Person[Id]$
$authorOf$	BP_{mo}	$AuthorPaper$	$PersonId$	$PaperId$	$AuthorPaper[PersonId] \subseteq Person[Id]$ $AuthorPaper[PaperId] \subseteq Paper[Id]$
$inProceed$	BP_{fo}	$Paper$	Id	$ProceedId$	$Paper[ProceedId] \subseteq Proceedings[Id]$

– if $P \in \mathsf{BP}_{mo}$, C is domain of P, D is range of P, then: $table(P) \neq R_C$, $table(P) \neq R_D$, $domCol(P) = A^d_P \in att(R_P)$, $rngCol(P) = A^r_P \in att(R_P)$, and $R_P.A^d_P \subseteq R_C.Id$, $R_P.A^r_P \subseteq R_D.Id$, where A^d_P is a foreign key referring from R_P to R_C and A^r_P is a foreign key referring from R_P to R_D, e.g., *authorOf*.

A mapping on schema level is specified by means of the following mapping rules (see source-to-target dependencies studied in [5,8]):

1. For each $C \in \mathsf{UP}_E$: $\forall x\ (C(x) \rightarrow \exists r\ (R_C(r) \wedge r.Id = x))$.
2. For each $P \in \mathsf{BP}_{fd}$, and $dom(P, C) \in \mathcal{R}$:
   $\forall x, y\ (P(x, y) \rightarrow \exists r\ (R_C(r) \wedge r.Id = x \wedge r.A^r_P = y))$.
3. For each $P \in \mathsf{BP}_{md}$, and $dom(P, C) \in \mathcal{R}$:
   $\forall x, y\ (P(x, y) \rightarrow \exists r, s\ (R_P(r) \wedge R_C(s) \wedge r.Id = x \wedge r.A^r_P = y \wedge s.Id = x))$.
4. For each $P \in \mathsf{BP}_{fo}$, $dom(P, C) \in \mathcal{R}$, and $rng(P, D) \in \mathcal{R}$:
   $\forall x, y\ (P(x, y) \rightarrow \exists r, s\ (R_C(r) \wedge R_D(s) \wedge r.Id = x \wedge r.A^r_P = y \wedge s.Id = y))$.
5. For each $P \in \mathsf{BP}_{mo}$, $dom(P, C) \in \mathcal{R}$, and $rng(P, D) \in \mathcal{R}$:
   $\forall x, y\ (P(x, y) \rightarrow \exists r, s, t\ (R_P(r) \wedge R_C(s) \wedge R_D(t) \wedge r.A^d_P = x \wedge s.Id = x \wedge r.A^r_P = y \wedge t.Id = y))$.

Note that the mapping rules above take into account also inclusion dependencies. Thus, it is guaranteed that the set of answers to a faceted query executed against the ontology is equal to the query evaluated in the relational database.

4 Faceted Search over Bibliography Ontology

Faceted search implies a new approach to modeling and perceiving data. It allows to see information objects in a multidimensional information space, like in multidimensional datawarehouse modeling. For example, $conferences$ can be perceived in a multidimensional space determined by such dimensions (called $facets$) as *time*, *location*, *authors* of papers, *publishers* of proceedings, etc.

However, in contrast to modeling in datawarehousing, where the distinction between target data (measures) and dimensions is fixed, in the case of faceted-oriented modeling this perception can change from query to query. For example, in one query the target information can be *persons* in a space determined by *conferences*, *papers* and *universities*. In another – *conferences* in a space determined by *persons* and *publishers*, etc.

An ontology, like that in Fig. 2, is in general a complex and large semantic network. Any unary predicate in this ontology can be treated as the target object. Then the others determine the multidimensional information space used to search the expected set of target objects. Thus, in one ontology can coexist many such information spaces.

To explore the ontology and utilize it to formulate queries, we implemented in DAFO an approach based on faceted search. In this implementation we distinguish the following three steps: (a) providing a faceted interface and initializing a faceted query by means of a keyword query, (b) refining the faceted query, (c) transforming the faceted query into an executable form.

4.1 Keyword Queries and Faceted Interfaces

A keyword query in DAFO is a partially ordered set of unary predicate names from the underlying ontology, $kq = (C_0, \ldots, C_N)$, $C_i \in \mathsf{UP}$, for $0 \leq i \leq N$, where C_0 is the type of expected answers (target objects). Elements in the sequence $(C_1, ..., C_N)$ are used to appropriate restricting and pivoting the ontology, and to arrange it in a hierarchy consistent with the ordering of unary predicates in the keyword query. This hierarchy forms a *faceted interface*, which is used by the user to iterative and interactive refinement of the faceted query.

In Figs. 3(a) and (b), there are two faceted interfaces determined by two different keyword queries. In Fig. 3(a), a user is interested in *Papers* presented at TPDL conferences and written by some persons not yet specified. In Fig. 3(b), a user is interested in *Persons* connected to some ACM or TPDL conferences.

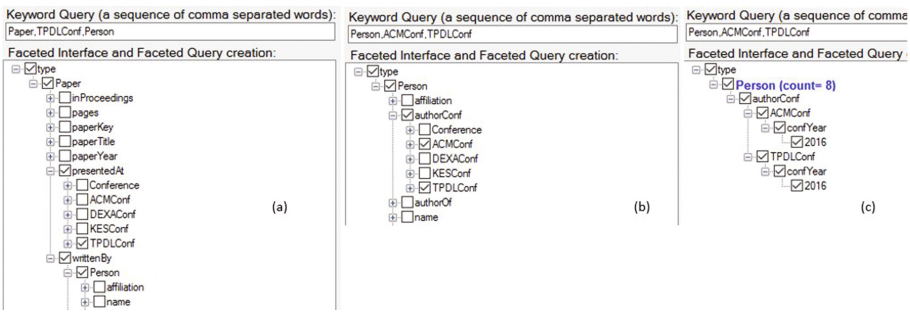

Fig. 3. Two faceted interfaces, (a) and (b), determined by two different keyword queries (checked nodes denote initial faceted queries), and a final form of faceted query (c), created over the interface (b).

In both cases, we have faceted interfaces, where checked elements form the first approximation of a created faceted query.

In both cases, the underlying ontology is pivoted in the way corresponding to the user intention expressed by the keyword query. Thus, the following two objectives are achieved: (1) the presentation of the ontology is restricted to some neighborhood of the given set of keywords (unary predicates); and (2) the ontology is somehow pivoted so that it is presented in the way conforming to the ordering of predicates in the keyword query.

A final faceted query in Fig. 3(c) is a result of operating over the faceted interface in Fig. 3(b).

4.2 Transforming Faceted Queries into First Order Faceted Queries

A faceted query is created over a faceted interface by means of selecting/unselecting nodes, inserting values of binary predicates, and discarding unselected nodes. During creation of faceted queries, the user is informed about the number of answers corresponding to the current form of the query.

The query in Fig. 3(c), has the following meaning:

> "*Get persons who are authors of papers presented at an ACM conference in year 2016, or at a TPDL conference in year 2016*".

The textual form of this query is:

$$\alpha = \{Person\}[(authorConf, \mathsf{any})/\{ACMConf[(confYear, \{\text{“2016”}\})], \\ TPDLConf[(confYear, \{\text{“2016”}\})]\}]. \quad (1)$$

The formal syntax of a faceted query α is [14,15]:

$$\begin{aligned} \alpha &::= t \mid t[\beta] \mid \alpha \vee \alpha \\ \beta &::= b \mid b/\alpha \mid \beta \wedge \beta, \end{aligned} \quad (2)$$

where: (a) t is a set $\{C_1, \ldots, C_n\}$ of unary predicates; (b) b is a pair (P, any) or $(P, \{a_1, \ldots, a_n\})$, where any denotes *any* constant, and $\{a_1, \ldots, a_n\}$ is a set of allowed constants (possible values of property P).

A faceted query with syntax (2) is transformed to a first order faceted query (FOFQ), which is in the class of MPEQs. The transformation is made by means of the following semantic function $[\![\alpha]\!]_x$:

$$\begin{array}{ll} [\![\{A_1, \ldots, A_n\}]\!]_x = A_1(x) \vee \cdots \vee A_n(x) & [\![t[b]]\!]_x = [\![t]\!]_x \wedge \exists y([\![b]\!]_{x,y}) \\ [\![\{a_1, \ldots a_n\}]\!]_x = (a_1 = x) \vee \cdots \vee (a_n = x) & [\![t[b/\alpha]]\!]_x = [\![t]\!]_x \wedge \exists y([\![b]\!]_{x,y} \wedge [\![\alpha]\!]_y) \\ [\![(R, \mathsf{any})]\!]_{x,y} = R(x, y) & [\![t[\beta_1 \wedge \beta_2]]\!]_x = [\![t[\beta_1]]\!]_x \wedge [\![t[\beta_2]]\!]_x \\ [\![(R, \{a_1, \ldots, a_n\})]\!]_{x,y} = R(x, y) \wedge [\![\{a_1, \ldots a_n\}]\!]_y & [\![\alpha_1 \vee \alpha_2]\!]_x = [\![\alpha_1]\!]_x \vee [\![\alpha_2]\!]_x. \end{array}$$

In result, a monadic positive existential query is obtained. For example, for the faceted query (1), we obtain:

$$\begin{aligned} [\![\alpha]\!]_x = Person(x) \wedge \exists x_1 (&authorConf(x, x_1) \wedge \\ &(ACMConf(x_1) \wedge \exists x_2 (confYear(x_1, x_2) \wedge x_2 = \text{“2016”}) \vee \\ &TPDLConf(x_1) \wedge \exists x_2 (confYear(x_1, x_2) \wedge x_2 = \text{“2016”}))). \end{aligned} \quad (3)$$

4.3 Rewriting FOFQs into Extensional Form

In FOFQs may occur both extensional and intentional predicates. In the rewriting process all intentional predicates are replaced with extensional ones using deductive rules from the ontology [13]. The rewriting algorithm recursively looks for intentional predicates. If C (or P) is such a predicate, then a rule with C (or P) occurring on its right-hand side is used in the rewriting procedure. The rewriting concerns the entire atom, i.e., $C(x)$ (or $P(x, y)$), and the atom is replaced by the left-hand side of the rule with appropriate substitution of variables. In result, some new intentional predicates can appear in the query, so the process of rewriting must be repeated recursively. If the set of rules is not recursive with respect to intentional predicates, and is complete, i.e., any intentional predicate occurs on the right hand side of some rule, then the rewriting process ends successfully.

For example, the atom containing intentional predicates $authorConf$ (see (3) and Fig. 4(b)) has the following rewriting (Fig. 4(c)) in ontology BibOn:

$$rewrite_{BibOn}(authorConf(x, x_1)) = \exists x_5(authorOf(x, x_5) \wedge Paper(x_5) \\ \wedge \exists x_6(inProceed(x_5, x_6) \wedge Proceedings(x_6) \wedge ofConf(x_6, x_1))). \quad (4)$$

In Fig. 4(a), we give a slightly modified version of the faceted query from Fig. 3(c) (requirements about affiliation of authors are added). The FOFQ before rewriting is presented in Fig. 4(b), and FOFQ after rewriting is in Fig. 4(c). Queries are depicted as syntactic trees, where all variables except x are quantified existentially. In Fig. 4(d) there is a sample set of answers to the query in DAFO.

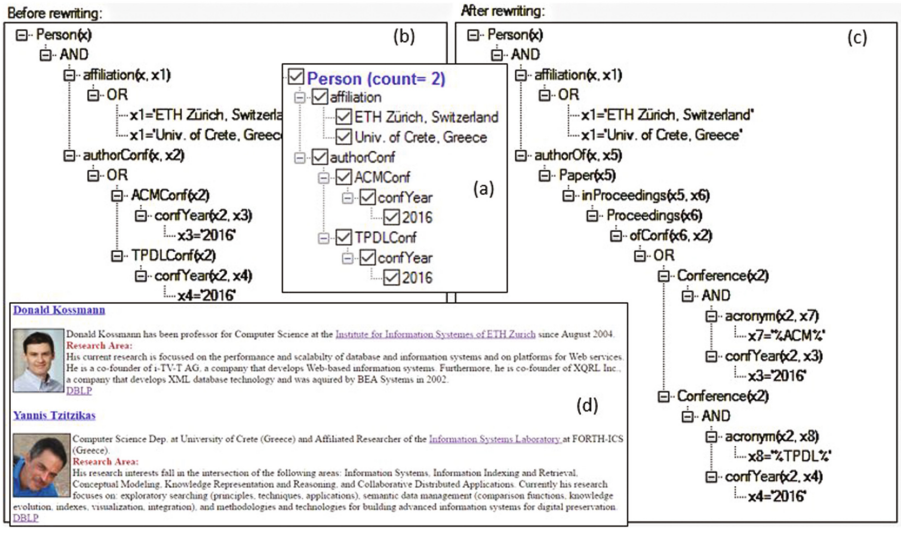

Fig. 4. Sample faceted query in DAFO (a), its presentation as: FOFQ tree before rewriting (b); FOFQ tree after rewriting (c); and answers to it (d).

4.4 Answering Faceted Queries - Experimental Evaluation

FOFQs are translated to SQL queries over relational database using the mapping defined in Sect. 3. A result SQL query is executed in relational databases using a commercial RDBMS (SQL Server, in this case). Advanced optimization capabilities provided by RDBMS guarantee high efficiency. This was verified in computational experiments made in DAFO system with the following setting: (a) a database containing: 3818 papers, 1907 conferences, 1853 proceedings, and 61 persons; (b) computation environment: 2.60 GHz Intel Core i7 processor, and 8GB RAM memory; (a) ontology with 182 elements (predicates and rules). Results of evaluations are given in Table 3 (query q3 is that in Fig. 4). Time costs (in milliseconds) are divided into *total preparing* and *execution* costs. The preparing time highly depends on both the size of query and ontology (in our experiments the ontology and the database were fixed).

Table 3. Evaluation of time costs for preparing and executing faceted queries.

Query	#Nodes after rewriting	Creation [msec]	Rewriting [msec]	Translation [msec]	Total preparing [msec]	Execution [msec]
q1	5	12	23	2	37	22
q2	9	6	58	8	72	29
q3	24	32	122	13	167	45
q4	37	53	210	16	279	57

We can observe that there is a linear relationship between the size of queries (expressed in the number of nodes in its syntactic tree after rewriting) and preparing and execution times. These relationships are presented in Fig. 5.

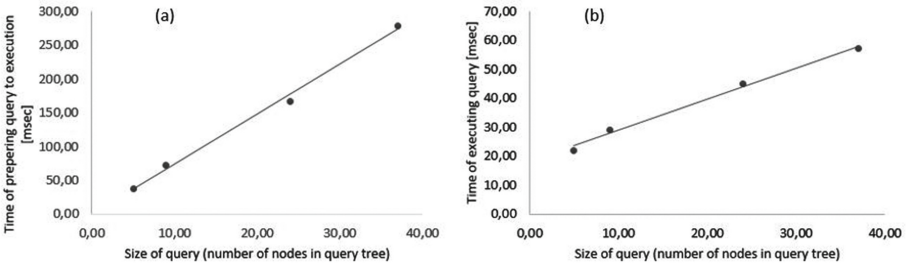

Fig. 5. Time of preparing faceted query to execution (a) and execution time (b) depending on the number of nodes in FOFQ tree after rewriting.

5 Summary

In this paper, we exploited the application of the ontology based data access (OBDA) approach equipped with faceted search utilities to explore bibliography databases. We discussed a way of using ontology to describe bibliography information space. Universality and flexibility of an ontology depends on the choice of deductive and integrity constraint rules. The former are used to deduce new facts and the latter to check correctness of data. Both are significant in rewriting queries. We proposed a way of presenting the ontology to users in conformance with the faceted search methodology. The implemented system DAFO provides users with a graphical interface allowing the user for interactive and iterative creation of faceted queries. We shown how the ontology can be mapped to a relational database. This mapping together with translation queries to SQL enables high efficiency of query execution. The crucial in achieving this efficiency was the usage of a commercial RDBMS with excellent optimization capabilities.

This research has been supported by Polish Ministry of Science and Higher Education under grant 04/45/DSPB/0163.

References

1. Arenas, M., Grau, B.C., Kharlamov, E., Marciuska, S., Zheleznyakov, D.: Faceted search over ontology-enhanced RDF data. In: ACM CIKM 2014, pp. 939–948. ACM (2014)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Petel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
3. Bak, J., Blinkiewicz, M.: RuQAR: Querying OWL 2 RL ontologies with rule engines and relational databases. In: 9th International Conference on Computational Collective Intelligence (ICCCI 2017), LNAI, Springer (2017). (in print)
4. Catarci, T., Costabile, M.F., Levialdi, S., Batini, C.: Visual query systems for databases: A survey. J. Vis. Lang. Comput. **8**(2), 215–260 (1997)
5. ten Cate, B., Kolaitis, P.G.: Structural characterizations of schema-mapping languages. Commun. ACM **53**(1), 101–110 (2010)
6. DBLP Computer Science Bibliography. http://dblp.org/ (2017)
7. Dumais, S.T.: Faceted search. In: Liu, L., Tamer Özsu, M. (eds.) Encyclopedia of Database Systems, pp. 1103–1109. Springer, Heidelberg (2009)
8. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci **336**(1), 89–124 (2005)
9. Krishnamurthi, S., Gray, K.E., Graunke, P.T.: Transformation-by-Example for XML. In: Pontelli, E., Santos Costa, V. (eds.) PADL 2000. LNCS, vol. 1753, pp. 249–262. Springer, Heidelberg (1999). doi:10.1007/3-540-46584-7_17
10. Ley, M.: DBLP - some lessons learned. PVLDB **2**(2), 1493–1500 (2009)
11. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Semant. **7**(2), 74–89 (2009)
12. OWL 2 Web Ontology Language Profiles. www.w3.org/TR/owl2-profiles (2009)
13. Pankowski, T.: Rewriting and executing faceted queries over ontology-enhanced databases. In: 21st Conference on Knowledge-Based and Intelligent Systems (KES 2017). Procedia Computer Science, Elsevier (2017, in print)

14. Pankowski, T., Brzykcy, G.: Data Access Based on Faceted Queries over Ontologies. In: Hartmann, S., Ma, H. (eds.) DEXA 2016. LNCS, vol. 9828, pp. 275–286. Springer, Cham (2016). doi:10.1007/978-3-319-44406-2_21
15. Pankowski, T., Brzykcy, G.: Faceted Query Answering in a Multiagent System of Ontology-Enhanced Databases. In: Jezic, G., Chen-Burger, Y.-H.J., Howlett, R.J., Jain, L.C. (eds.) Agent and Multi-Agent Systems: Technology and Applications. SIST, vol. 58, pp. 3–13. Springer, Cham (2016). doi:10.1007/978-3-319-39883-9_1
16. Papadakos, P., Tzitzikas, Y.: Hippalus: Preference-enriched faceted exploration. In: Workshops of the EDBT/ICDT. CEUR Workshop Proceedings, vol. 1133, pp. 167–172. CEUR-WS.org (2014)
17. Tunkelang, D.: Faceted Search. Morgan & Claypool Publishers, San Rafael (2009)
18. Zloof, M.M.: Query-by-example: A data base language. IBM Syst. J. **16**(4), 324–343 (1977)