DAFO: an ontological database system with faceted queries

Tadeusz Pankowski and Jarosław Bąk

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

Abstract. We present an ontology-based data access provided with a faceted interface. The underlying relational database is perceived through an ontological schema. Queries against the ontology are formulated using a hierarchical faceted interface. We show main ideas of DAFO and discuss how it relates to well-known OBDA systems.

1 Introduction

DAFO (*Data Access based on Faceted queries over Ontology*) [6, 7] is an ontological database system, where the conceptual schema is specified as an ontology and the extensional part is stored in a relational database. Queries in DAFO are formulated by end-users by means of an interactive graphical faceted interface. Queries expressible in DAFO are equivalent to concept expressions in description logic (DL):

$$\text{DL-PE} \subseteq \text{DL-GN} \subseteq \text{DL-GN}^{\theta k},$$

where DL-PE is a class of *positive existential* queries, DL-GN is a class of queries with *guarded negation*, and DL-GN$^{\theta k}$ is an extension of DL-GN with *number restriction* (*count aggregation restriction*) [2]. Description logic is a natural base for faceted queries since queries in both cases are monadic and tree-shaped [1].

In OBDA (*Ontology-Based Data Access*) systems [8], queries are formulated in SPARQL, and mappings are defined from a relational database schema to the ontological schema using GAV (Global As View) approach. As a consequence, the unfolding based on mappings can be used in query rewriting. Like in OBDA, DAFO ontology is based on OWL 2 QL. In DAFO, we follow LAV (Local As View) approach, i.e., the ontological schema is mapped to an underlying relational database schema. Then the translating queries into SQL is more complex than in OBDA but ensures more flexibility. In faceted-based approach [5], a user can gradually refine her/his query using an interactive faceted query interface. The challenging issues are then: (a) presenting the relevant part of the underlying ontological schema in a form of a hierarchical faceted interfaces, (b) expressiveness of faceted queries which are formulated using only click operations on the faceted interface, (c) rewriting faceted queries into SQL.

Further on, we will describe and illustrate some main features of DAFO referring to a running example based on the example discussed in [3].

2 Ontology in DAFO

In Figure 1, there is a sample *Staff* ontology (see [3]) specifying an application domain of a project company. Nodes are labeled by types (unary predicates), and edges are labeled by object properties (binary predicates). Data properties are neglected in the figure. Extensional predicates (solid lines) are those that are mapped to a relational database, while intensional (dotted lines) are defined by means of some rules.

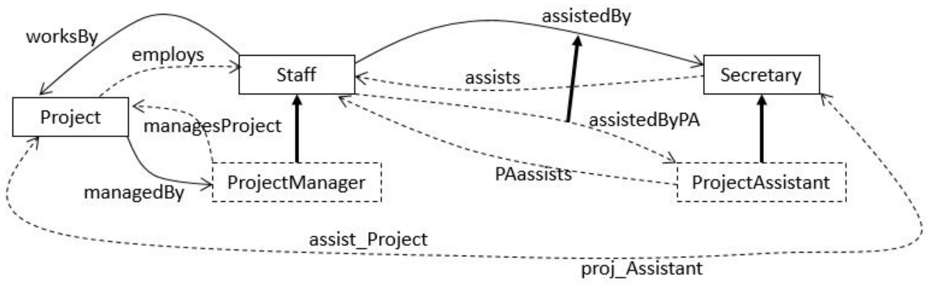


Fig. 1. *Staff* ontology graph: solid and dotted lines denote extensional and intensional predicates, respectively; fat arrows denote subsumptions between types and properties.

Domains and ranges of binary predicates, as well as subtype and subproperty relationships, can be deduced from the graph. Some other rules are shown below:

$(R1)\ \forall x\ (ProjectManager(x) \to \exists y\ assistedByPA(x, y))$
$(R2)\ \forall x\ (\exists y\ managesProject(x, y) \to ProjectManager(x))$
$(R3)\ \forall x, y\ (managedBy(x, y) \leftrightarrow managesProject(y, x))$
$(R4)\ \forall x, y, z\ (managedBy(x, z) \wedge assistedBy(z, y) \to proj\_Assistant(x, y))$

3 Query formulation and rewriting in DAFO

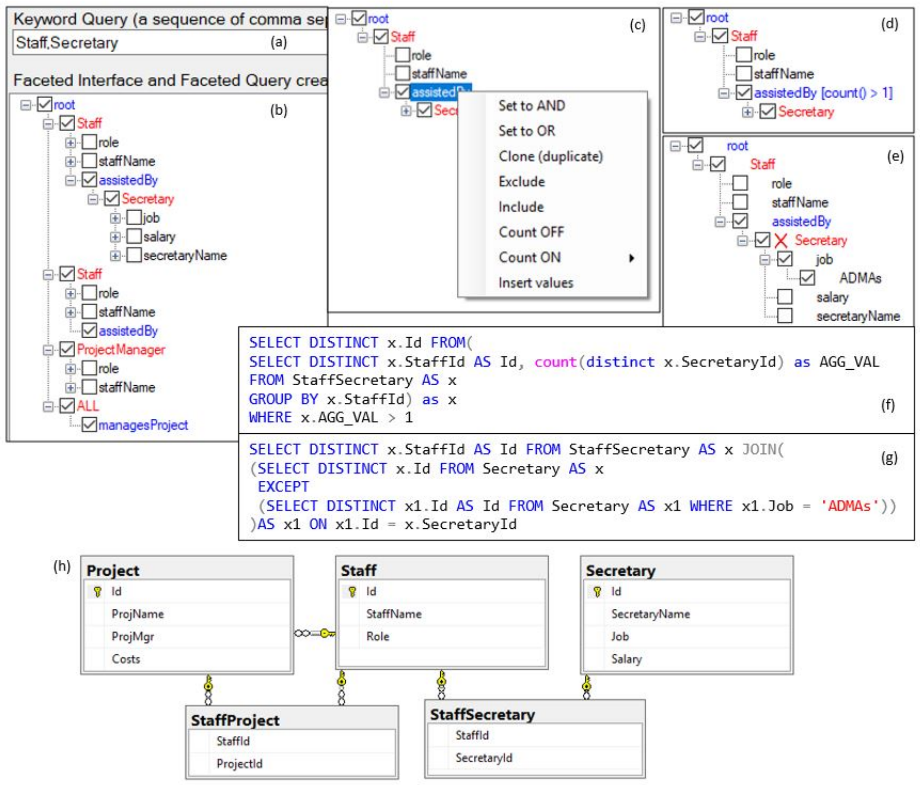
Assume that we are interested in staff members assisted by secretaries. In DAFO, instead of trying to write a query, a user starts with providing a sequence of types (a *keyword query*). The first type in the sequence denotes the type of answers, and the others are intended to be used in defining restrictions (in faceted queries, Figure 2(e)). In this case, the keyword query is (*Staff*, *Secretary*), Figure 2(a).

The reasoning engine in DAFO uses the set of rules in the ontology to find all possible associations between *Staff* and *Secretary*. In result, an initial query $q(x)$ is produced and displayed in a form of faceted interface, Figure 2(b). The query, denoted by checked nodes in Figure 2(b), is the union of monadic conjunctive tree-shaped queries (of the class DL-PE):

$$\begin{aligned} q(x) = {} & Staff(x) \wedge \exists y\ assistedBy(x, y) \wedge Secretary(y) \vee \\ & Staff(x) \wedge \exists y\ assistedBy(x, y) \vee \\ & ProjectManager(x) \vee \exists y\ managesProject(x, y) \end{aligned} \tag{1}$$

$ProjectManager(x)$ is in $q(x)$ in force of $(R1)$ stating that $assistedByPA$ is total on $ProjectManager$ (i.e., on a subset of its domain) so, every project manager must be assisted by at least one project assistant. $\exists y\ managesProject(x, y)$ is in $q(x)$ since, in force of $(R4)$, is a subset of $ProjectManager$.

A user can accept the query, uncheck some nodes or refine the query by performing some operations on the interface displayed in the context menu in Figure 2(c). The query in Figure 2(d) asks for members of staff who are assisted by more than one secretary (query of the class DL-GN$^{\theta k}$). A SQL form of this query is shown in Figure 2(f). The query in Figure 2(e) asks for members of staff who are not assisted by secretaries with job equal to *'ADMAs'* (query of the class DL-GN). The corresponding SQL query is in Figure 2(g). In Figure 2(h), there is a relational database schema the *Staff* ontology is mapped to.



```
SELECT DISTINCT x.Id FROM(
SELECT DISTINCT x.StaffId AS Id, count(distinct x.SecretaryId) as AGG_VAL
FROM StaffSecretary AS x
GROUP BY x.StaffId) as x
WHERE x.AGG_VAL > 1
```

```
SELECT DISTINCT x.StaffId AS Id FROM StaffSecretary AS x JOIN(
(SELECT DISTINCT x.Id FROM Secretary AS x
 EXCEPT
 (SELECT DISTINCT x1.Id AS Id FROM Secretary AS x1 WHERE x1.Job = 'ADMAs'))
)AS x1 ON x1.Id = x.SecretaryId
```

Fig. 2. Screenshots of DAFO system during operating on faceted interface (a)-(d), SQL forms of faceted queries (f)-(g), and schema of the underlying relational database (h).

Like in OBDA, queries in DAFO are processed in two stages: (a) the *ontology-mediated query* (OMQ) rewriting is used to produce a first-order query, (b) the query is translated into SQL. In general, the rewriting in OBDA produced by the rewriting algorithm `PerfectRef` [4] can be prohibitively large and complex [8]. For example, query (1) seems to be unnecessarily complex. Indeed, the second disjunction in $q(x)$ subsumes the first one. Techniques used in query rewriting depend on the assumption concerning preservation of consistency in the underlying database. Because of a dynamic character of the system, some updates may not be propagated making the database inconsistent. Moreover, due to the presence of rules like ($R1$), the database can be incomplete. These facts must be taken into account in the rewriting process. In consequences, the final query can have a large number of unions. If we assume that the underlying database is consistent, i.e., all ontology rules are satisfied, we can use optimization techniques based on conjunctive query containment. Then the final form of query (1) can be simpler, like that in Figure 2(c). In DAFO, both alternatives are available.

4 Summary

We presented DAFO system, which combines: (a) a user friendly, yet expressive, faceted-oriented approach to query formulation over ontologies, (b) usage of an ontology to semantic enrichment of relational database, and (c) utilization of the mature relational DBMS engine offering advanced optimization capabilities to store data (the extensional part of the ontology) and answer queries.

Acknowledgement. We acknowledge the support from the grant 04/45/DSPB/0185.

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 (2003)
3. Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. J. ACM 65(5), 28:1–28:51 (2018)
4. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. J. Autom. Reason. 39(3), 385–429 (2007)
5. Ferré, S., Hermann, A.: Semantic search: Reconciling expressive querying and exploratory search. In: The Semantic Web - ISWC 2011. LNCS, vol. 7031, pp. 177–192. Springer (2011)
6. Pankowski, T.: Exploring ontology-enhanced bibliography databases using faceted search. In: Research and Advanced Technology for Digital Libraries - TPDL 2017. LNCS, vol. 10450, pp. 27–39. Springer (2017)
7. Pankowski, T.: Schema transformations and query rewriting in ontological databases with a faceted interface. In: Theory and Practice of Model Transformation - ICMT 2018. LNCS, vol. 10888, pp. 76–91. Springer (2018)
8. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: 27-th Joint Conference on Artificial Intelligence, IJCAI 2018. pp. 5511–5519 (2018)