Towards Query Formulation and Query-Driven Ontology Extensions in OBDA Systems

B. Cuenca Grau[2], M. Giese[4], I. Horrocks[2], T. Hubauer[3], E. Jiménez-Ruiz[2], E. Kharlamov[2], M. Schmidt[1], A. Soylu[4], D. Zheleznyakov[2]

[1] fluid Operations AG, Germany
[2] Department of Computer Science, University of Oxford, Oxford UK,
[3] Siemens Corporate Technology, Germany
[4] Department of Informatics University of Oslo, Norway

Abstract. The process of translating end-users' information needs into executable and optimised queries over the data is the main problem that end-users face in Big Data scenarios. In this paper we present the recently started EU project Optique, which advocates for a next generation of the well known *Ontology-Based Data Access* (OBDA) approach to address this problem. We discuss challenges, present ongoing work, and our current preliminary solutions with regards to the query formulation and query-driven ontology extension.

Keywords: Ontology-Based Data Access, Query Formulation, Ontology Navigation

1 Introduction

Massive amounts of data have been accumulated over decades; moreover, data keeps increasing fast; and it is spread over a vast variety of formats and sources, being modeled using different conceptualisations of the domain, often using schemata that are optimized for efficient processing rather than for intuitive access. These three aspects go hand in hand with the *volume*, *velocity*, and *variety* dimensions of Big Data [7].

Accessing the *relevant* data in this context is becoming increasingly difficult for end-users. For example, in large enterprises, such as Statoil,[5] end-users work with applications that allow accessing data through a limited set of predefined queries. In situations where an end-user needs data that these predefined queries do not provide, the help of IT-experts (e.g., database managers) is required. The IT-experts need to translate the end-users' information needs into suitable queries and this process may require several iterations. In particular in the oil and gas industry, IT-experts spend 30–70% of their time gathering and assessing the quality of data [5]. This is clearly very expensive in terms of both time and money. The Optique project[6] [7] advocates for the well-known *Ontology-Based Data Access* (OBDA) approach (e.g., [20, 2]) to address the bottlenecks that end-users face when accessing Big Data and aims at solutions that significantly reduce the cost of data access.

[5] Statoil ASA is an oil and gas company and it is one of the uses case scenarios in Optique, which aims in particular at providing access to data for geologists, petrophysicists, etc.

[6] `http://www.optique-project.eu/`

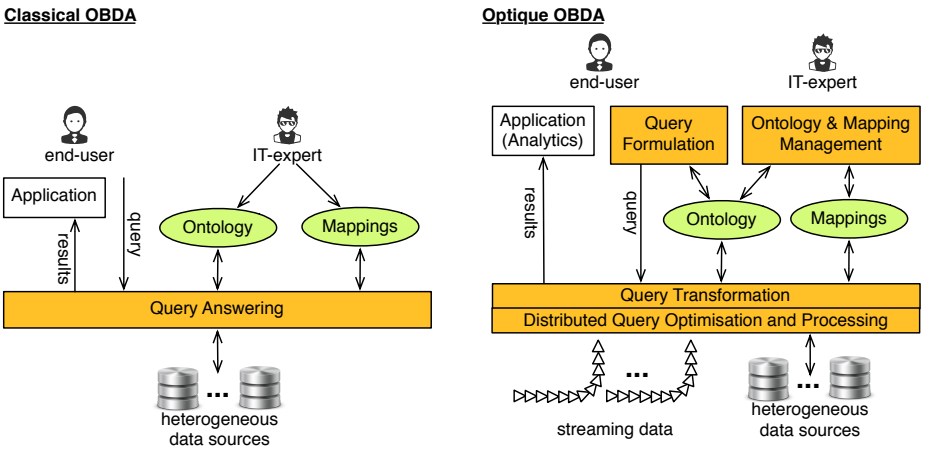

Fig. 1. The general architecture of a classical (left) and the Optique (right) OBDA system

OBDA systems have the potential to address the data access problem by presenting a general ontology-based and end-user oriented query interface over heterogeneous data sources. The core elements in a classical OBDA system (Figure 1, left) are an *ontology*, which describes the application domain in terms of user-oriented vocabulary of classes (usually referred as concepts) and relationships between them (usually referred as roles), and a set of *mappings*, which relates the terms in the ontology and the schema of the underlying data sources. End-users formulate queries using the terms defined by the ontology, which should correspond to their view of the domain, and thus, they are not required to understand the data source schemata. For example, in the Statoil use case the ontology would provide concepts such as WellBores, their purpose, etc., while the mappings would associate SQL queries to each term of the ontology vocabulary, i.e., similarly to SQL view definitions. For example, the ontology concept Water_Injection_Wellbore would be mapped to the SQL query:

```
SELECT *
FROM   DevelopmentWellBore
WHERE  purpose=Injection and content=Water
```

To be precise, one should extend this mapping with a reification function that transforms tuples returned by the query above into constants, i.e, exact identifiers of water injection wellbores. Another alternative is to adjust the query by changing the select clause to the following: **SELECT** ID.

State-of-the-art OBDA systems that are based on classical OBDA architecture (Figure 1, left), however, have shown among others the following four limitations.

1. The *usability* of OBDA systems regarding the user interface is still an open issue. Even if the vocabulary provided by the ontology is familiar to end-users, they may find difficult to formulate complex queries when several concepts and roles are involved.

2. OBDA systems critically depend on a *suitable ontology* and the corresponding set of *mappings*, which are in practice expensive to obtain. Even if we assume that the ontology and the mappings are given, they are not static artifacts and should evolve according to the new end-users' information requirements. Both bootstrapping of ontologies and mappings for an initial installation of OBDA systems and subsequent maintenance are challenging topics which are still in a premature stage.
3. Treatment of query answering is usually limited to query rewriting and there is little support of *distributed* query optimisation and processing in OBDA systems.
4. *Streaming*, e.g., sensor, data and corresponding *analytical tools* are generally ignored by OBDA systems, which seriously limits their applicability in enterprises such as Statoil.

The Optique project, which started in November 2012 and has a four years time span, aims at addressing these four limitations by developing a next generation OBDA system that targets the demands of today's Big Data challenges. The core components of the Optique's OBDA solution are presented in Figure 1, right: *(i)* query formulation, *(ii)* ontology and mapping management, *(iii)* query transformation, and *(iv)* distributed query optimisation and processing. Besides the core components, Optique's OBDA system integrates both data streams and databases, and supports data analytics.

In this paper we focus on the first and partially the second limitation above. More specifically, we focus on the query formulation component of Optique's solution. We will also discuss the query-driven ontology extension sub-component of the query formulation component which in fact partially addresses issues of the second limitation (ontology maintenance). In the following sections we discuss challenges, introduce our ongoing work, and illustrate our preliminary solutions. Moreover, we present an envisaged architecture of our query formulation component.

2 Challenges in Query Formulation

The ontology in an OBDA system, as already mentioned, is intended to provide a user-oriented conceptual model of the domain. This allows users to formulate queries using familiar terms and shields from understanding the structure of the underlying data sources. However, in order to provide the necessary power and flexibility, the required query language will inevitably be rather complex and it would be unrealistic to expect all end-users to formulate queries directly in such a query language.

In Optique we advocate for a *query by navigation* (QbN) approach combined with faceted search to address the usability problem. We refer interested readers to [23, 17, 10, 21] for some state-of-the-art solutions. There are, however, two important conceptual challenges related to the query by navigation approach: *(i)* representation paradigms for ontologies, and *(ii)* correlations between navigation and query construction. We will now elaborate on these challenges.

Representation paradigms. Query by navigation approaches usually combine navigational search and faceted search techniques over an underlying ontology graph (or any other kind of structured knowledge). Thus, in this scenario, the ontology not only provides the domain vocabulary but also guides the end-user to formulate complex queries.

Existing approaches, however, are mostly dominated with one type of representation paradigm (e.g., forms, diagrams etc.), hence limited to the confines of a particular model. We believe that multiple representation paradigms should be used in collaboration where each paradigm is responsible for the tasks for which it is best suited.

We have also observed that current solutions do not adequately employ a very important paradigm, namely the *graph representation metaphor* of OWL ontologies. The formal underpinning of OWL (and its revision OWL 2) is provided by Description Logics (DLs) where the fundamental modelling concept is an *axiom* (i.e., a logical statement relating roles and/or concepts). This is a key difference from traditional graph-based knowledge representation paradigms (e.g., semantic networks). OWL ontologies may include complex axioms and concept constructors such as universal restrictions that do not have a direct representation in a graph structure.

Correlations between navigation and query construction. Given a navigation paradigm, one has to understand how the actual navigation influences the construction of a query. More precisely, how the navigation corresponds to operators in a given query language. For example, *how to form a query with negation, disjunction, or aggregation via a graph navigation?* For navigation over graphs there is a natural correspondence to conjunctive queries: moving along a graph can be seen as an extension of the corresponding query with more conjuncts corresponding to the (labels of) edges and nodes met on the way. This correspondence gives good opportunities for designing QbN algorithms. For other types of queries, however, establishing the correlation is a challenging problem that will require further research.

The representation paradigms and the correlation between navigation and query construction give two dimensions of choices for query by navigation approaches. Orthogonally, the ontology and query languages give another two dimensions to choose from. In the following we elaborate on the OWL 2 QL ontology language and queries that essentially correspond to a conjunctive fragment of SPARQL. For this choice we will discuss possible issues and challenges.

OWL 2 QL and conjunctive queries. Even in the simplified scenario where the ontology language is reduced to the OWL 2 QL profile [19], and only conjunctive queries are formulated, there are still several issues regarding to the representation and navigation of the information shown to the user:

1. *Top-down propagation of property restrictions*. Traditional graph representations usually only include explicit information attached to a concept in the ontology; however, inherited restrictions will also play an important role in graph navigation. For example if the ontology includes the axiom $\mathsf{Wellbore} \sqsubseteq \exists \mathsf{hasPath}.\mathsf{Path}$,[7] then the subconcepts of Wellbore should also suggest a link to the concept Path. However, this can make the representation unfeasible when Wellbore has many subconcepts; thus a trade-off between readability and the amount of necessary information provided to the user should be achieved.
2. *Bottom-up propagation of property restrictions*. Since from a model-theoretic point of view the interpretation of an OWL concept also includes the interpretations of all

[7] The axiom says that every wellbore has (at least) one path.

its subconcepts, it may also make sense to suggest for a given concept the (potential) restrictions of its subconcepts. For example, consider an ontology including GasWell and OilWell as (direct or indirect) subconcepts of Well and the axioms OilWell $\sqsubseteq \exists$hasProduction.Oil and GasWell $\sqsubseteq \exists$hasProduction.Gas, then the concept Well could potentially be related to the concepts Oil and Gas.

3. *Cycles in the ontology graph*. Ontology axioms such as inclusion between concepts or inverse roles can lead to cycles in the ontology graph. Thus, the navigation should take into account these cycles and, in some cases, avoid repetitive suggestions. For example, if one constructs a query by navigating through the following ontology and starts the navigation from the concept Wellbore, then one gets back to Wellbore in two steps, via the concepts Core and StratigraphicLayer.

$$\begin{aligned} \text{Wellbore} &\sqsubseteq \exists \text{hasCore.Core} \\ \text{Core} &\sqsubseteq \exists \text{hasLayer.StratigraphicLayer} \\ \text{StratigraphicLayer} &\sqsubseteq \exists \text{layerOf.Wellbore}. \end{aligned}$$

   Should the system suggest or allow the user to go to Wellbore *via the* layerOf *relation when* StratigraphicLayer *is reached?* The answer depends on the query that the user has in mind. For example, if the user has the following query in mind,[8] then Wellbore should be recommended.

$$\begin{aligned} \text{Q(x)} \text{ :- } & \text{Wellbore(x), hasCore(x,y), hasLayer(y,“Neolithic”),} \\ & \text{layerOf(“Neolithic”,u), Wellbore(u).} \end{aligned}$$

   Since the way to cope with cycles depends on the user's intention, we do not envision generic solutions to this problem. At the same time, it is useful to notify users when they are confronted by cycles and to provide them with some form of control, e.g., by restricting the depth of constructed queries or by allowing recursion.
4. *Negative information*. Negative information such as disjointness between concepts should be exploited accordingly. For example, if the end-user selects the wells with oil as a production type and the concepts OilWell and GasWell are disjoint in the ontology, then the navigation system could safely skip suggestions related to GasWell.
5. *Role inclusion axioms* will also lead to extra complexity when navigating over the ontology graph. For example, consider the axioms BottomHoleAssembly $\sqsubseteq$ $\exists$hasBit.DrillBit and hasBit $\sqsubseteq$ hasPart, then the concept DrillBit should also be suggested as *a part of* BottomHoleAssembly.

In Optique we intend to design and implement novel techniques that take into account the issues above. We aim at providing an intuitive end-user interface while preserving the semantics of the underlying ontology in order to formulate both complex and valid queries. In particular, we intend to look at existing work, where query formulation is driven from a Description Logic model of the domain, e.g., [1, 4].

[8] This query is written in the Datalog notation

2.1 Query-driven ontology extensions

The ontology may not include all the vocabulary expected or needed by the end-user. Moreover, the vocabulary is to a certain extent specific to individuals, projects, departments, etc. and subject to change. Thus, keeping the ontology up-to-date with respect to the end-user needs arises as an indirect (but crucial) challenge in query formulation. In Optique we differentiate the following changing scenarios driven by end-user information requirements:

1. *Adding new synonyms*. Concept synonyms (e.g. annotation labels) do not represent new logical extension of the ontology, and hence end-users will be able to add them to the ontology with no (logical) harm. For example, the concept WellBore can be extended with the labels "drill hole" or "borehole". In order to avoid an overloading of the ontology with synonyms, we advocate a separation between the ontology (e.g. logical axioms) and the terminological information (e.g. synonyms, descriptions, related terms, etc.) as proposed in [14].
2. *Adding basic extensions*. End-user queries may also require basic extension of the ontology hierarchy, such as adding a new concept GeologicalWellBore under WellBore (i.e. GeologicalWellBore $\sqsubseteq$ WellBore). These types of additions can be considered *safe* since they represent a *conservative extension* of the ontology [12]. However other additions to the ontology may require further analysis by the IT-expert if they are not conservative extensions (e.g. reclassifying the concept WellBore under the new concept PlannedSideTrack).
3. *"On the fly" extensions*. This represents the more challenging scenario where we intend to exploit ontology learning techniques in order to mine formulated queries and to identify relevant new concepts and relations (e.g., [24, 16]). Ontology alignment techniques (e.g. [11]) will also be required in order to relate the new vocabulary to the existing ontology concepts.
4. *IT-expert assistance*. In the cases where the manual or on-the-fly extensions are insufficient, the assistance of the IT expert will be required to extend the ontology accordingly.

3 Envisaged Architecture and Approach

Figure 2 shows the main query formulation components envisaged for the Optique OBDA solution and their interaction with other components of the system. Next we give a brief overview of each of them. Note that many components deal with both one-time queries, e.g., SPARQL queries, and continuous queries, e.g., CSPARQL queries.

1. *Editing components*. Different users may cooperate on the same query or set of queries, thus, the Optique solution aims at providing (at least) three kind of interfaces to formulate the query (i.e. components): *(i)* direct editing, *(ii)* context sensitive editing, and *(iii)* query by navigation exploiting faceted search and other navigation paradigms. Technically versed users may prefer the direct editing of the query using a formal language (e.g. SPARQL, stream query language), while other end-user should be provided with a less technical interface such as query by navigation. Additionally, direct editing should also allow the possibility of exploiting

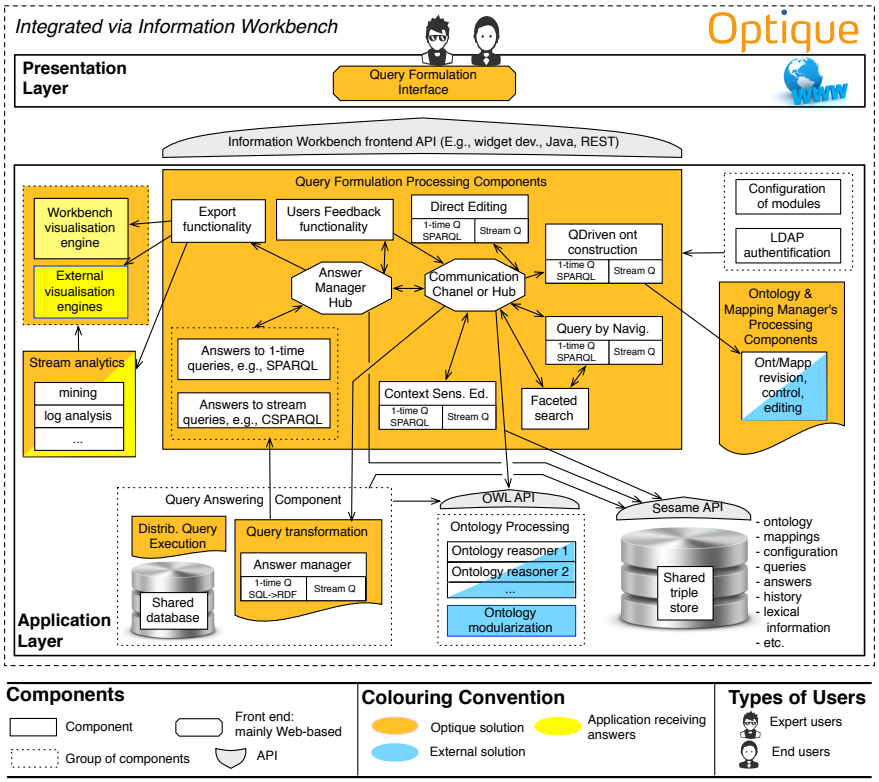

Fig. 2. Query Formulation components of the Optique OBDA system

the ontology, and provide context sensitive completion. All three interfaces should provide views on the partially constructed query, and users should be able to switch between views at will.

2. *Query-driven ontology extension component* will manage the ontology extensions driven by the query requirements and will send the new ontology versions to the Ontology Revision Control component for further analysis and validation of the performed changes.

3. *The Ontology Processing component*. The ontology will be a key element for the query formulation component and thus, the ontology processing component (e.g. OWL API, OWL reasoners) will also play an important role. Furthermore, logic-based ontology modularization techniques [6] will also be exploited to achieve a good balance between overview and focus when dealing with large ontologies. The properties of such modules guarantee that the semantics of the concepts of interest are preserved while providing (in general) a much smaller fragment of the ontology.

4. The *The Query Answering component* will transform the formulated queries into executable and optimized queries with respect to the data sources (e.g. streaming data, relational databases).

5. *The Answer Manager component*. This component should deal with the (basic) visualization of the query results and their transformation (i.e. *export functionality*) into the required output formats (e.g. input formats of external Statoil tools).
6. *The User Feedback component*. This component is intended to allow the user to semi-automatically refine a query if the (partially) obtained results are not the expected ones. Furthermore, similar or related queries to the partially constructed query will also be suggested in order to help end-users in the refinement.
7. *The Ontology Revision Control component*. Different versions of the ontology may exist concurrently (e.g. extensions driven by different formulated queries or query requirements). These versions will be managed by the IT-experts through a revision control system in order to detect logical defects (e.g. unsatisfiabilities), logical conflicts among versions as in [13], and OWL 2 profile violations (e.g. a new version is outside the OWL 2 QL profile).

All components will be integrated into the Information Workbench [9, 8], a generic platform for semantic data management, which provides a central triple store for managing the OBDA system assets (such as ontologies, mappings, etc.), generic interfaces and APIs for semantic data management, and a flexible user interface that will be used for implementing the query formulation components. The user interface follows a semantic wiki approach, based on a rich, extensible pool of widgets for visualization, interaction, mashup, and collaboration, which can be flexibly integrated into semantic wiki pages, allowing developers to compose comprehensive, actionable user interfaces without any programming effort. The following subsection presents the technical architecture for the query formulation interface and the solution approach based on widget-based mashups.

3.1 Widget-based solution

A mashup based approach (cf. [22]) is promising for the construction of an extensible and flexible query formulation interface. The mashup idea, in our context, is grounded on the possibility to combine the functionality and data of a set of individual applications in a common graphical space, for common tasks. Widgets are the building blocks of mashups, where each widget corresponds to a standalone application with less complex functionality and presentation compared to full-fledged applications. In query formulation scenario, a set of widgets can be employed, for instance, one for query by navigation and one for faceted search for handling the construction of queries; and one for representing results in table and one for visualizing the result in a graph to handle communication of results to the end-users.

Widgets are managed by a widget environment which provides basic communication and persistence services to widgets. The orchestration of widgets relies on the requirement that each widget discloses its functionality to the environment through a client side interface and notifies any other widget in the environment (e.g., broadcast, subscription etc.) and/or the widget environment upon each user action. Then, either each widget decides on what action to execute in response, by considering the syntactic or semantic signature of the received event; or, the environment decides which widgets to invoke with which functionality. The core benefits of such an approach are that,

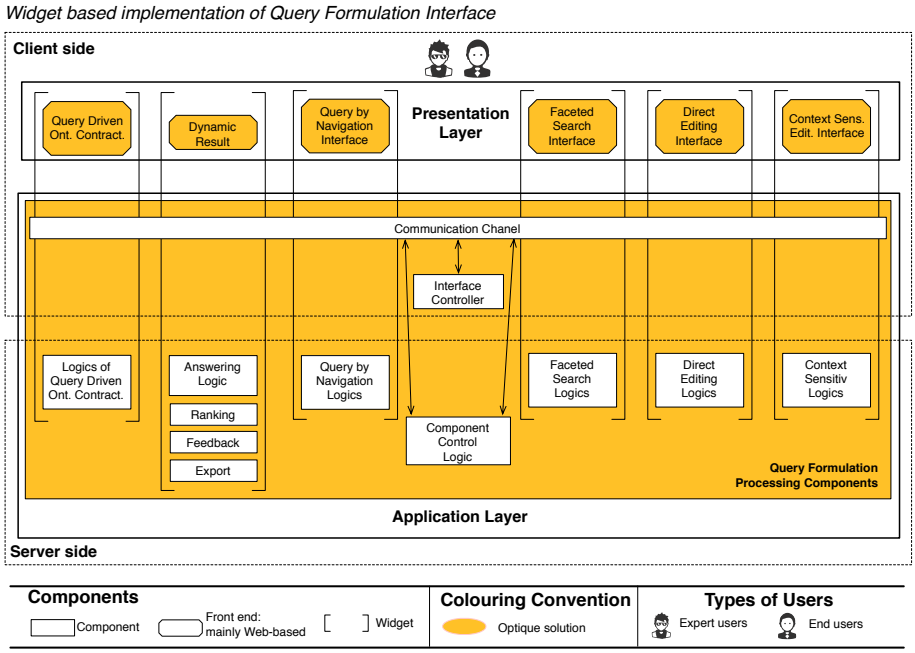

Fig. 3. Query Formulation interface based on widget-based mashups

i it becomes easier to deal with the complexity, since the management of functionality and data can be delegated to different widgets;

ii each widget can employ a different visualization paradigm that best suits the functionality that it is expected to provide;

iii widgets can be used alone or together, in different combinations, for different contexts and experiences; and,

iv the functionality of the overall interface can be extended by introducing new widgets (e.g., such as for result visualization).

A possible architecture for a query formulation interface based on widget-based mashups is depicted in Figure 3. The architecture assumes that each widget has client side and server side components (for complex processing), and that widgets can communicate with each other and with the environment through a communication channel. Communication usually happens through the client side, but a server side communication mechanism can also be realized in order to support remote experiences (e.g., while widgets running on remote devices). The architecture assumes that there exists an environment controller at the client side and a component control logic at the server side. The former is responsible for operational tasks such as collecting the event notifications from widgets and submitting control commands to them. The latter is responsible for the orchestration logic, that is it decides how widgets should react to specific events.

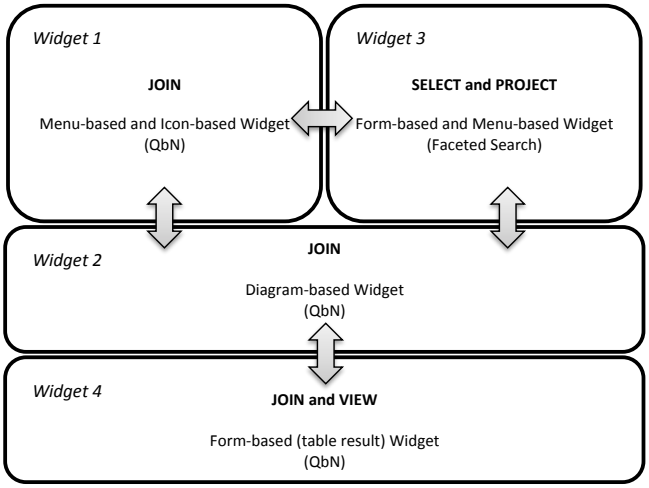

Fig. 4. An initial approach combining different paradigms for query formulation

3.2 Query formulation interface

Catarci et al. [3] categorize data access efforts into *understanding the reality of interest* (i.e., exploration), which relates to activities for finding and understanding schema concepts and relationships relevant to information need; and *query construction*, which concerns the compilation of relevant concepts and constraints into formal information needs. The query construction task is normally considered as a series of actions, each of which can be either a *select*, *join*, or *project* action. The join type of actions enables users to combine different concepts and to form path expressions for queries, where the select and project type of actions allow users to specify the properties that are to be returned and to impose constraints to filter the results. As such, the choice of visual representation and interaction paradigm, along with underlying metaphors, analogies etc., is of primary importance for the query formulation interface.

We have observed that a single representation and interaction paradigm is not sufficient for developing a successful query formulation interface. Therefore, we strive to combine the best parts of different paradigms (cf. [18]). A conceptual sketch of our first attempt is shown in Figure 4. Initially, there are four widgets available. The first widget is based on a menu-based approach with QbN interaction paradigm, where domain concepts, properties, and relationships are distributed into a set of layers, with respect to a certain hierarchy or organization, and presented in the form of lists. This widget also employs an icon-based paradigm by supplementing domain vocabulary with meaningful icons. The second widget follows a diagram-based approach with QbN. The diagram-based approach utilizes geometric symbols to depict relationships among schema concepts. The third widget employs a form-based and menu-based approach in the form of a faceted search interface. The form-based approach adopts conventional paper forms as a metaphor. The final widget is also form-based, more specifically table-based, and employs a QbN based interaction style.

The first widget is responsible for join actions, and determines the focus of interface. First, available domain concepts are shown to the user; as soon as a user selects a domain concept, the selected concept becomes the focus, and relationships pertaining to this concept are listed. The second widget is responsible for providing an overview by allowing the user to switch between a graph visualization of the query and the ontology. The third widget presents the properties of the focus concept in the form of fields and menu-items to enable the user to select properties of interest and to specify constraints on them. The fourth widget represents query results in a common table view and enables user to navigate at instance level by accessing other facts that are linked to the result items in the table view.

The proposed approach provides a good balance between view and overview and supports domain exploration and query construction efforts. It also provides an ample amount of room for supportive features, since it is typically not possible to address every requirement with visual representations [15]. This particularly becomes true for large ontologies, in which guiding the user to relevant vocabulary is of crucial importance. For instance, a keyword search facility can support finding relevant concepts, properties and relationships in the first and second widgets. Each representation paradigm can handle different ontology axioms, for instance, a faceted search paradigm is better suited for representing disjointness, and a menu-based paradigm with QbN may be a better option for handling cycles (e.g., with path coloring).

4 Conclusions

We have presented the main challenges to be faced in the design and development of the query formulation and query-driven ontology extension solutions. Although the EU project Optique is still in an early stage, we aim at turning our preliminary ideas into novel solutions in the very near future, and to evaluating their effectiveness in our industry use cases. This will provide us with invaluable feedback to inform ongoing research and development of enhanced query formulation components.

Acknowledgements. The research presented in this paper was financed by the Seventh Framework Program (FP7) of the European Commission under Grant Agreement 318338, the Optique project. Cuenca Grau, Horrocks, Jiménez-Ruiz, Kharlamov, and Zheleznyakov were also partially supported by the EPSRC projects ExODA and Score!

References

1. Bechhofer, S., Horrocks, I.: Driving User Interfaces from FaCT. In: Proceedings of the 2000 International Workshop on Description Logics. pp. 45–54 (2000)
2. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web 2(1), 43–53 (2011)
3. Catarci, T., Costabile, M., Levialdi, S., Batini, C.: Visual query systems for databases: A survey. Journal of Visual Languages and Computing 8(2), 215–260 (APR 1997)

4. Catarci, T., Dongilli, P., Mascio, T.D., Franconi, E., Santucci, G., Tessaris, S.: An ontology based visual tool for query formulation support. In: ECAI. pp. 308–312 (2004)
5. Crompton, J.: Keynote talk at the W3C Workshop on Semantic Web in Oil & Gas Industry: Houston, TX, USA, 9–10 December (2008), available from http://www.w3.org/2008/12/ogws-slides/Crompton.pdf
6. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. 31, 273–318 (2008)
7. Giese, M., Calvanese, D., Haase, P., Horrocks, I., Ioannidis, Y., Kllapi, H., Koubarakis, M., Lenzerini, M., Möller, R., Özçep, O., Rodriguez Muro, M., Rosati, R., Schlatte, R., Schmidt, M., Soylu, A., Waaler, A.: Scalable End-user Access to Big Data. In: Rajendra Akerkar: Big Data Computing. Florida : Chapman and Hall/CRC. To appear. (2013)
8. Haase, P., Hütter, C., Schmidt, M., Schwarte, A.: The Information Workbench as a Self-Service Platform for Linked Data Applications. In: the WWW 2012 Developer Track (2012)
9. Haase, P., Schmidt, M., Schwarte, A.: The Information Workbench as a Self-Service Platform for Linked Data Applications. In: Proceedings of the Second International Workshop on Consuming Linked Data (COLD) (2011)
10. Heim, P., Ziegler, J.: Faceted visual exploration of semantic data. In: Second IFIP WG 13.7 conference on Human-computer interaction and visualization. pp. 58–75 (2011)
11. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: Int'l Sem. Web Conf. (ISWC). pp. 273–288 (2011)
12. Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., Berlanga, R.: Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: The 5th European Semantic Web Conference, ESWC. vol. 5021, pp. 185–199 (2008)
13. Jiménez-Ruiz, E., Grau, B.C., Horrocks, I., Llavori, R.B.: Supporting concurrent ontology development: Framework, algorithms and tool. Data Knowl. Eng. 70(1), 146–164 (2011)
14. Jimeno-Yepes, A., Jiménez-Ruiz, E., Llavori, R.B., Rebholz-Schuhmann, D.: Reuse of terminological resources for efficient ontological engineering in life sciences. BMC Bioinformatics 10(S-10), 4 (2009)
15. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods - A survey. ACM Computing Surveys 39(4) (2007)
16. Kotis, K., Papasalouros, A., Maragoudakis, M.: Mining query-logs towards learning useful kick-off ontologies: an incentive to semantic web content creation. IJKEDM 1(4) (2011)
17. Lim, S.C.J., Liu, Y., Lee, W.B.: Faceted search and retrieval based on semantically annotated product family ontology. In: Proc. of teh Workshop on Exploiting Semantic Annotations in Information Retrieval. pp. 15–24 (2009)
18. Lohse, G., Biolsi, K., Walkner, N., Rueter, H.: A classification of visual representations. Communications of the ACM 37(12), 36–49 (DEC 1994)
19. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles (2009), W3C Recommendation
20. Rodriguez-Muro, M., Calvanese, D.: High Performance Query Answering over DL-Lite Ontologies. In: the 13th Int'l Knowledge Representation and Reasoning Conf. (KR) (2012)
21. Soylu, A., Modritscher, F., De Causmaecker, P.: Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. Integrated Computer-Aided Engineering 19(1), 93–109 (2012)
22. Soylu, A., Modritscher, F., Wild, F., De Causmaecker, P., Desmet, P.: Mashups by orchestration and widget-based personal environments Key challenges, solution strategies, and an application. Program-Electronic Library and Information Systems 46(4), 383–428 (2012)
23. Suominen, O., Viljanen, K., Hyvänen, E.: User-Centric Faceted Search for Semantic Portals. In: Proc. of the 4th European Semantic Web Conf. (ESWC 2007). pp. 356–370 (2007)
24. Zhang, J., Xiong, M., Yu, Y.: Mining query log to assist ontology learning from relational database. In: Frontiers of WWW Research and Development (APWeb). pp. 437–448 (2006)