Semantic Web 0 (0) 1–35
IOS Press

OptiqueVQS: a Visual Query System over Ontologies for Industry [1]

Editor(s): Freddy Lecue, Accenture Tech Labs, Ireland
Solicited review(s): Vanessa Lopez, IBM Research, Ireland; Two anonymous reviewers

Ahmet Soylu [a,*], Evgeny Kharlamov [b], Dmitriy Zheleznyakov [b], Ernesto Jimenez-Ruiz [c], Martin Giese [c], Martin G. Skjæveland [c], Dag Hovland [c], Rudolf Schlatte [c], Sebastian Brandt [d], Hallstein Lie [e] and Ian Horrocks [b]

[a] *Department of Computer Science, Norwegian University of Science and Technology, Gjøvik, Norway*
E-mail: ahmet.soylu@ntnu.no
[b] *Department of Computer Science, University of Oxford, Oxford, UK*
E-mail: {evgeny.kharlamov, dmitriy.zheleznyakov, ian.horrocks}@cs.ox.ac.uk
[c] *Department of Informatics, University of Oslo, Oslo, Norway*
E-mail: {ernestoj, martingi, martige, hovland, rudi}@ifi.uio.no
[d] *Corporate Technology, Research and Technology Center, Siemens AG, Munich, Germany*
E-mail: sebastian-philipp.brandt@siemens.com
[e] *Statoil ASA, Stavanger, Norway*
E-mail: hali@statoil.com

Abstract. An important application of semantic technologies in industry has been the formalisation of information models using OWL 2 ontologies and the use of RDF for storing and exchanging application data. Moreover, legacy data can be virtualised as RDF using ontologies following the ontology-based data access (OBDA) approach. In all these applications, it is important to provide domain experts with query formulation tools for expressing their information needs in terms of queries over ontologies. In this work, we present such a tool, OptiqueVQS, which is designed based on our experience with OBDA applications in Statoil and Siemens and on best HCI practices for interdisciplinary engineering environments. OptiqueVQS implements a number of unique techniques distinguishing it from analogous query formulation systems. In particular, it exploits ontology projection techniques to enable graph-based navigation over an ontology during query construction. Secondly, while OptiqueVQS is primarily ontology driven, it exploits sampled data to enhance selection of data values for some data attributes. Finally, OptiqueVQS is built on well-grounded requirements, design rationale, and quality attributes. We evaluated OptiqueVQS with both domain experts and casual users and qualitatively compared our system against prominent visual systems for ontology-driven query formulation and exploration of semantic data. OptiqueVQS is available online and can be downloaded together with an example OBDA scenario.

Keywords: visual query formulation, OWL 2 ontologies, RDF data, SPARQL queries, data retrieval, usability

1. Introduction

Adoption of semantic technologies has been a recent development in many large companies such as IBM [34], the steel manufacturer Arcelor Mittal [5], the oil and gas company Statoil [51], and Siemens [1, 55,74]. An important application of these technologies

[1] This work was funded by the EU FP7 Grant "Optique" (agreement 318338); the EPSRC projects MaSI[3], DBOnto and ED[3]; the BIGMED project (IKT 259055); and the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project no.: 237889).
[*] Corresponding author. E-mail: ahmet.soylu@ntnu.no

has been the formalisation of *information models*[1] using OWL 2 ontologies and the use of RDF for storing application data. OWL 2 provides a rich and flexible modelling language, which is well-suited for describing industrial information models [2,35,54]. It comes with unambiguous and standardised semantics, as well as a wide range of tools to develop, validate, integrate, and reason with such models. In turn, RDF data can not only be seamlessly accessed and exchanged, but also stored directly in highly scalable RDF triple stores and effectively queried in conjunction with the available ontologies. Moreover, legacy and other data that must remain in its original format and cannot be transformed into RDF can be virtualised as RDF using ontologies following the *ontology-based data access* (OBDA) approach [58,70,94].

In all these applications, it is important to provide domain experts—who have extensive domain knowledge but not necessarily skills and knowledge in semantic technologies and formal query languages such as SPARQL—with query formulation tools for expressing their *information needs* over ontologies. The problem of query formulation for end users has been acknowledged by many [7,21,36,91] and numerous systems have been developed so far. These systems can be categorised as follows:

1. *Textual query editors* (e.g., the Virtuoso SPARQL Query Editor[2]) employ the full expressivity of SPARQL, but demand technical skills and knowledge (i.e., on syntax and schema). Context-aware editors, such as SparQLed [19], offer auto-completion and recommendations based on the schema and dataset.
2. *Keyword search* (e.g., [14]) interprets a query as a bag of words. These systems are simple to use, but are inherently limited in expressiveness. There are approaches, such as KESOSD [62] and SWSE [42], that aim at increasing the accuracy and completeness of keyword search.
3. *Natural language interfaces* (e.g., [47,61]) interpret natural language phrases as queries, taking linguistic considerations into account, but suffer from ambiguities and linguistic variability. There are approaches to overcome this problem, such as user dialogues for feedback and clarification [26].
4. *Visual query languages* (VQL), such as RDF-GL [43] and QueryVOWL [38], are based on a well-defined formal semantics with a visual notation and syntax. They are comparable to formal textual languages as they demand technical skills and knowledge to interpret the visual formalism.
5. *Visual query systems* (VQS) [21], such as Rhizomer [15] and Konduit VQB [4], are based on a system of interactions rather than a visual formalism, and therefore can have a design demanding no or limited technical background. They often compromise expressivity to reach a fine balance between expressiveness and usability.

To the best of our knowledge, no system from any of these categories has been developed to meet industrial requirements or evaluated with industrial users. In this work we present a VQS, namely OptiqueVQS [86,89], which is designed upon *(i)* requirements from Statoil and Siemens consolidated during the joint OBDA project Optique[3] [31,32], and *(ii)* best HCI practices for interdisciplinary engineering environments.

OptiqueVQS implements a number of unique techniques that distinguish it from comparable query formulation systems. In particular:

- it exploits ontology projection techniques to enable graph-based navigation over an ontology during query construction;
- while OptiqueVQS is primarily ontology driven, it exploits sampled data to enhance selection of data values for some data attributes;
- it is built on well-grounded requirements, design rationale, and quality attributes; and
- it is evaluated with different types of end users in different contexts.

We evaluated OptiqueVQS with different user groups and contexts: a study involving casual users [89]; a comparative study with PepeSearch (a form-based query interface) [103]; and three studies with Statoil and Siemens domain experts reported in this article. Our studies provided encouraging results; in particular, studies with Statoil and Siemens users revealed that domain experts could use OptiqueVQS to formulate queries meeting their daily data needs in a few minutes with high effectiveness and efficiency.

[1] An information model is a representation of concepts and the relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse [59], such as functionality of and information flow between different assets in a power plant [54,72].

[2] See e.g. `http://dbpedia.org/sparql` for an instance of the Virtuoso editor.

[3] Optique project: `http://optique-project.eu`

Finally, we qualitatively compared OptiqueVQS against prominent existing visual systems for ontology-driven query formulation and exploration of semantic data that are the most relevant to our system. For the comparison, we considered gFacet [41], OZONE [95], SparqlFilterFlow [37], Konduit VQB [4], Rhizomer [15], PepeSearch [102], Super Stream Collider framework [73], and TELIOS Spatial [28]. The comparison revealed that OptiqueVQS possesses an important set of quality attributes relevant in an industrial context, while others meet only a few of them. OptiqueVQS is available online and can be downloaded together with an example OBDA scenario including a data set, an ontology, mappings etc. from the project's website (see Section 6 for details).

The rest of this article is organised as follows: in Section 2, we highlight the main contributions of this article with links to our related work. In Section 3, we present preliminary notations and concepts used throughout the article. In Section 4, we present the Statoil and Siemens use cases. In Section 5, we discuss a set of requirements and quality attributes, while in Section 6, we present OptiqueVQS itself. In Sections 7 and 8, we evaluate OptiqueVQS, first against the requirements and then using a set of usability studies. In Section 9, we discuss related work and similar tools, while in Section 10 we discuss both our main findings and the limitations of OptiqueVQS. Finally, we conclude the article and discuss the future work in Section 11.

2. Contributions

OptiqueVQS can be used in two different scenarios – see Figure 1: directly over a triple store through an endpoint (i.e., scenario A), or over an OBDA framework that virtualises relational data into RDF (i.e., scenario B). In both scenarios, the setup consists of two parts. Part A deals with *visual query formulation* and Part B with *query answering*. The contributions of this article fall into visual query formulation (i.e., part A), while OBDA is the application scenario (i.e., scenario B) [32].

Our work builds on several iterative and self-standing focused research activities complementing each other. This article extends our previous work on OptiqueVQS in several important directions. In particular, we present:

(a) a detailed account of the Statoil and Siemens use cases, characterising existing data sources, data access infrastructure, end-users, and data access routines such as the frequency of interaction, variance of query tasks, and structural complexity of query tasks;
(b) requirements collected systematically, including a comprehensive analysis and classification of representative queries collected from use cases and established best practices culled from experience with visual query systems for relational databases;
(c) a qualitative comparison with eight other query formulation systems based on quality attributes collected from literature and a set of corresponding features realising the suggested quality attributes;
(d) a detailed analysis of design choices behind OptiqueVQS supporting the design rationale behind each representation and interaction paradigm and extensions for spatial and temporal query formulation support;
(e) three extensive user studies with domain experts at Siemens and Statoil and a discussion of limitations and key findings from these studies; and
(f) an improved and extended OptiqueVQS backend presented in detail, including its expressive power and the role of and motivation behind each individual component.

Regarding the links to our previous and related work, extensions (a), (b), (c), and (d) follow the methodology and a qualitative analysis of visual query formulation approaches that we extracted through a survey [80,91]. We discussed initial ideas on design and implementation of OptiqueVQS with domain experts at Statoil and Siemens (d) by rapidly developing a prototype [85,86].

An early user study with casual users on a generic domain [89] provided us with early insights before experimenting with the domain experts (e); — domain experts often have constrained availability, while casual users are more accessible and have desired characteristics for our purposes (i.e., lack of technical knowledge and skills). Additionally, a comparative user study [103] allowed us to compare the suitability of graph and form-based paradigms for different user groups (d) and revealed supporting evidence for our core design choice. The OptiqueVQS backend [88,89] has been consolidated continuously as a result of the requirements collected through user experi-

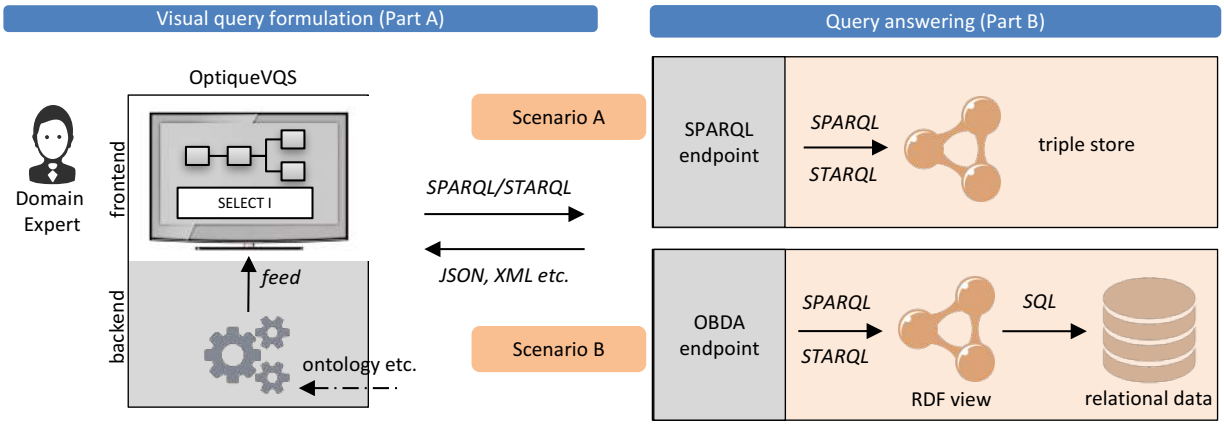

Fig. 1. OptiqueVQS in context – possible scenarios and architectural overview.

ments (f). For example, we developed a technique for ranking and ordering ontology elements (i.e., adaptive query formulation) based on query logs (f), which falls outside the scope of this article but is presented elsewhere [87].

The Optique solution, as an end-to-end OBDA framework including OptiqueVQS, has been deployed in Statoil [51,53,56] and Siemens [50,52,57]. The specifics of OBDA setup, such as mappings, query transformation, deployment, and query answering (i.e., the part B of scenario B) are also beyond the scope of this article.

3. Preliminaries

In the following, we give a brief tour of some notions from description logics (DL), RDF, OWL, SPARQL queries and their semantics. The goal of this section is to semi-formally introduce some relevant semantic web notions that we follow, in order to prepare the reader for the examples and explanations appearing in the article. Since in this article we study how to support the construction of queries over ontologies in industrial settings and focus mostly on user driven requirements rather than complexity and other formal accounts, we make the formal descriptions below light weight and refer the reader to relevant material for more details.

We use standard notions from first-order logic. We assume pairwise disjoint countably infinite sets of *constants*, *unary predicates* (also called *classes* or *atomic classes*) and *binary predicates* (also called *properties*). Constants, in turn, are divided into disjoint sets of *objects* (also called *individuals*) and *literal values*. A *fact* is a ground atom and a *knowledge base* is finite set of facts.

An *ontology* is a finite set of first-order sentences. The web ontology language OWL 2 [24] is a recursive set of ontologies, closed under renaming of constants and the subset relation. Each OWL 2 ontology can be represented using a specialised DL syntax [8,45] where variables are omitted and which provides operators for constructing complex concepts and properties from simpler ones, as well as a set of axioms. The semantics of an OWL 2 ontology is defined in a standard way using first-order interpretations [64]. Note that, for convenience and readability, in the examples we use the Manchester OWL syntax [44], which is a user-friendly compact syntax for OWL 2 ontologies

For example, consider the following statement about the application domain:

"every wellbore has (at least) one core".

This statement can be written as an OWL 2 axiom of the form (see Section 6.2.1 for further details):

Wellbore SubClassOf: hasCore some Core

SPARQL [40] is the standard query language to access RDF data. SPARQL queries are defined in terms of *basic graph patterns* (BGPs) $Q(\vec{x})$, i.e., sets of triples of the form $\langle n_1, e, n_2 \rangle$, where the n_i denote nodes in a graph and e denotes an edge, and all three can either be concrete nodes (i.e., constants or unary predicates) and edges (i.e., binary predicates), or variables; some of these variables form the vector $\vec{x}$ of Q's

output variables. In this work, we focus on the construction of SPARQL queries where basic graph patterns do not have variables on the second position, nor on the third position when e is rdf:type. This means that, we do not allow predicates as variables, and thus our queries can naturally be represented as conjunctions of unary and binary atoms. SPARQL 1.1 also supports filtering, union of BGPs, aggregate functions, and other operators. We allow only a limited number of these due to usability concerns. The semantics of query answering is defined in the standard way in terms of homomorphism [3]: a vector of constants $\vec{t}$ is the answer for a conjunctive query $Q(\vec{x})$ over a dataset D, if there is a homomorphism from the query to D such that the vector of output variables is matched to $\vec{t}$. These semantics can be naturally extended from datasets to *first-order logic* (FOL) interpretations of datasets and ontologies [8].

4. Industrial Use Cases

In the context of the Optique project, the development and evaluation of OptiqueVQS was guided by use cases from Statoil and Siemens, including sample queries and data sets. In both cases, domain experts typically access relevant data through predefined queries. IT experts, who have extensive technical knowledge but often lack domain knowledge, extract relevant data through an *extract-load-transform* (ETL) process [29] when an information need is not met through available queries. However, this approach is quite inefficient and highly iterative due to *miscommunication between IT experts and domain experts*, *high workload of IT experts*, *complexity of query formulation*, and *long query execution times*.

Statoil and Siemens have their data stored in relational databases rather than triple stores, as majority of the world's enterprises do. In the Optique project, the use case data sets have been represented as knowledge bases using an OBDA technology to enable in-place querying of legacy relational data sources [32]. OBDA technologies are important in the context of visual query formulation as well, as they extend the reach of ontology-based visual query formulation from triple stores to relational databases; hence, raising it as a viable and realistic solution for all. The OBDA approach we employed is built on two mechanisms [18,22,94]:

(a) *mappings* declaratively relate ontological terms with the underlying data and are used to virtualise the relational data in databases into graph data expressed over a language defined in an ontology;
(b) and *query rewriting* is used to expand and translate the posed queries (e.g., SPARQL) into the language of the underlying relational database system (e.g., SQL) hosting the data for execution.

The specifics of the OBDA framework are out of scope of this work; therefore, we refer interested readers to the Optique project [31,32,50,51].

In the following subsections, we describe the characteristics of each use case. We believe that they are representative for many of the data access challenges faced by today's data-intensive industries. Descriptions were provided by the organisations themselves and confirmed through interviews and on-site visits. We highlight and mark some parts of the descriptions as evidences (i.e., ^{En}) supporting the requirements presented in Section 5. Due to corporate confidentiality policies, some numbers in the descriptions have been approximated.

4.1. Statoil use case

The overall goal of the Statoil use case in the Optique project is to enable *geologists and geoscientists at Statoil to find answers to their information needs —questions that generally concern locating new petroleum deposits — and there are currently 900 geologists and geophysicists in Statoil accessing data routinely* $^{(E1)}$. Domain experts with technical skills and knowledge are rare. There are several complex and large databases and schemas in use are typically designed from an abstract generic information model and present themselves quite obscurely to their end users. *For example, one of the databases, called EPDS, currently has about 3,000 tables with about 37,000 columns. Building interesting SQL queries require therefore in many cases a very large number of table joins (i.e., 50 to 200 joins)* $^{(E2)}$, thus making the task of handcrafting SQL queries towards this database a very complex and time-consuming process.

To access the data sets, Statoil personnel use special purpose software tools that contain predefined and mostly generic queries. The data sets are never directly accessed by domain experts through handwritten queries. Hence, in order to answer specific and detailed information needs a Statoil domain expert must gather data from the answer sets of multiple such

predefined queries and process the answers by manipulating, joining and filtering the data in other software tools, like spreadsheet applications. This is a manual task that is prone to error, inefficient and difficult to automate and reproduce. Moreover, the database extraction tool is complex and *contains a large number of predefined queries* (E3), so finding the correct queries can be an elaborate process. Due to the complexity of the tool and the underlying database schema new queries are in practice never added to the tool. *Domain experts spend considerable time on data extraction activities daily in oil & gas industry; in Statoil 30-70% of time spent on analytics task is used for extracting the right data* (E4); therefore, in this situation, the value creation potential is severely limited.

4.2. Siemens use case

Siemens runs several service centres for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components such as generators and compressors. *Diagnosis engineers working at the service centres are informed about any potential problem detected on site* (E5). Unlike Statoil, a good number of diagnosis engineers at Siemens have technical skills and knowledge. They access a variety of raw and processed data with predefined queries in order to isolate the problem and to plan appropriate maintenance activities. For diagnosis situations not initially anticipated, new queries are required, and an IT expert familiar with both the power plant system and the data sources in question has to be involved *to formulate various type of queries; in Siemens there are around 4,000 predefined queries and query patterns. On average, 35 queries are modified monthly, 10% of queries are modified yearly, and several new queries are added every month* (E6). Thus, unforeseen situations may lead to significant delays of up to several hours or even days.

The required data is *spread over hundreds of tables with very complex structure for event data; the database size is in the order of hundreds of terabytes growing at a rate of 30 GB per day coming from appliances (up to 2,000 sensors in each) and static data sources* (E7). With few built-in features for manipulating time intervals, traditional data base systems offer insufficient support for querying time series data, and it is highly non-trivial to combine querying techniques with the statistics-based methods for trend analysis that are typically in use in such cases. *Domain experts' daily routines are very data intensive, and they spend 80% of their diagnostic time on gathering the relevant data* (E8). IT experts will not be required anymore for adding new queries, and manual pre-processing steps can be avoided by enabling domain experts to formulate complex queries on their own with respect to an expressive and high-level domain vocabulary.

5. Requirements

Domain experts have in-depth knowledge and understanding of the semantics of their domain of expertise. However, they might or might not have technical skills and knowledge on programming, databases, and query languages. In the latter case, they often have low tolerance, intention, or time to use and learn formal textual query languages. Therefore, our primary goal is to provide a visual query specification mechanism for users who cannot or do not desire to use formal textual query languages to retrieve data. We also expect that domain experts with technical skills and knowledge could often benefit from the availability of such visual mechanism, particularly if they are given the opportunity to switch between textual and visual query formulation within a task.

Visual query formulation [21,80,91] as an *end-user development* paradigm [60] is promising to remediate the end-user data access problem. It is built on the *direct manipulation* idea [77], in which end users recognise and interact with the visual representations of domain elements, rather than recalling domain and syntax elements and programmatically combining them. Epstein [30] considers visual approaches for query formulation in two categories, VQS and VQL, which we introduced earlier. However, a VQL, compared to a VQS, demands considerable technical skills and knowledge to interpret the visual semantics and syntax and to understand the relevant technical jargon.

A VQS has to support certain data access efforts: *exploration*, i.e., understanding the reality of interest, which relates to the activities for understanding and finding schema concepts and relationships relevant to the information need at hand; and *construction*, which concerns the compilation of relevant concepts and constraints into formal information needs (i.e., queries) [21]. On these grounds, the choice of *visual representation* and *interaction paradigms* along with the underlying metaphors, analogies etc. is of primary importance. Catarci et al. [21] classify VQSs with respect to both visual representation paradigms such as

forms, diagrams and icons, and interaction paradigms such as navigation and browsing. The choice of appropriate representation and interaction paradigm depends on query, task, and user types, e.g. variance of query tasks, structural complexity of queries, and users' familiarity with the subject domain [21].

One should also realise the distinction between *browsing* and *querying*. Browsing means that users, to a large extent, operate at data level to filter down an information space using e.g. faceted search interfaces (e.g., [99]). When querying, which we predominantly use in OptiqueVQS, users directly interact with the vocabulary of the domain (concepts and relations) (e.g., [11]), but not directly with concrete data as e.g. in OLAP cube interfaces. This is necessary because:

(a) the queries we need to pose are more complex than what can be achieved by more data oriented interfaces;
(b) and both the evaluation of those queries and the caching of all possible precomputed results would use too much resources.

For domain experts and other non-skilled users, query formulation is a complex task; therefore, an end-user visual query formulation tool is often limited in expressiveness to ensure good usability. End users make very little use of advanced functionalities and are likely to drop their own requirements for the sake of having simpler ways for basic tasks [20]. But even a VQS at the right level of expressiveness is not necessarily adopted by end users and organisations unless it also reaches a certain level of quality in terms of *user experience*, *system design*, and *run-time performance*.

Overall, we highlight three main challenges:

(C1) Identifying common query types (i.e., *typicality*) that are reasonably complex (i.e., *perceived complexity*) and would meet the majority of end-users' information needs to set an appropriate balance between usability and expressiveness.
(C2) Identify query, task and user types at hand in order to select representation and interaction paradigms that fit best.
(C3) Identify a set of *quality attributes* [48], i.e., non-functional requirements, ensuring that a VQS can function and evolve as needed.

In the following, we list an elaborate set of requirements in terms of expressivity and quality attributes.

5.1. Expressiveness

In order to address C1, we first studied the typicality by constructing a query catalogue from 97 representative sample queries provided by Statoil in natural language. Information needs in the query catalogue are considered as patterns of information needs, and each such request represents one topic that geologists are typically interested in. We verified with domain experts that the catalogue provides a good coverage for the information needs of Statoil geologists.

Two SPARQL experts reformulated these information needs in SPARQL given a domain ontology. Then we made a syntactical analysis of the query catalogue (see Figure 2) with respect to notable query types described in Table 1 and with respect to the SPARQL specification [40]. These query types include conjunctive queries (QT1), disjunctive queries (QT2), queries with cycles (QT3), queries with aggregation (QT4), queries with negation (QT5), and ground queries (QT6). The identification of queries of QT1, QT2, QT4 and QT5 are straight forward as they are built on clear SPARQL operators; however, identification of QT3 queries is more involved as it relates to the topology of a given query. Therefore, we transformed each query into an undirected-labelled graph and executed a cycle detection algorithm to identify QT3 queries.

The analysis suggests that the majority (64%) of Statoil's queries are ground queries – see Figure 2 (a). A similar analysis of queries supplied by Siemens later in natural language revealed that a large part (40%) of Siemens' queries are ground queries – see Figure 2 (b). These analyses are in line with the literature suggesting that many user queries are tree-shaped conjunctive queries [69] – i.e., conjunctive queries without cycles. Considering perceived complexity by the end users, we assumed queries including cycles and queries including disjunction and negation, particularly at object property level, to be comparatively harder. The first group requires visiting the same node twice in a query, while the second group requires a deeper understanding of these notions. Therefore, we are led to the first requirement:

(R1) Support the formulation of tree-shaped conjunctive queries.

In order to address C2, we conducted a thorough conceptual literature survey [91]. Particularly the long-standing literature on visual query formulation over

Table 1
Description of query types.

#	Query type	Description	SPARQL syntax
QT1	Conjunctive queries	Query atoms in a given query are connected only with AND connective.	SPARQL queries with basic graph patterns (BGP) and group graph patterns (GGP).
QT2	Disjunctive queries	Some query atoms in a given query are connected with OR connective.	SPARQL queries with multiple optional graph patterns (MOGP), i.e. OPTIONAL, and alternative graph patterns (AGP), i.e., UNION.
QT3	Queries with cycle	Query graph includes at least one path where a node is visited twice.	SPARQL queries having at least one path staring and ending with the same node, when the query graph is viewed as undirected labelled graph.
QT4	Queries with aggregation	The values of multiple output elements are grouped together to form a single value.	SPARQL queries including aggregate functions such as MIN, MAX, AVG, SUM.
QT5	Queries with negation	Queries that involve checking whether certain triples don't exist in the data graph.	SPARQL queries that involve negation by failure through NOT EXISTS, MINUS, NOT IN, and !BOUND operators.
QT6	Ground queries	Queries that are conjunctive and tree-shaped and do not include negation and aggregation.	SPARQL queries that are conjunctive and do not include cycle, aggregation, and negation.

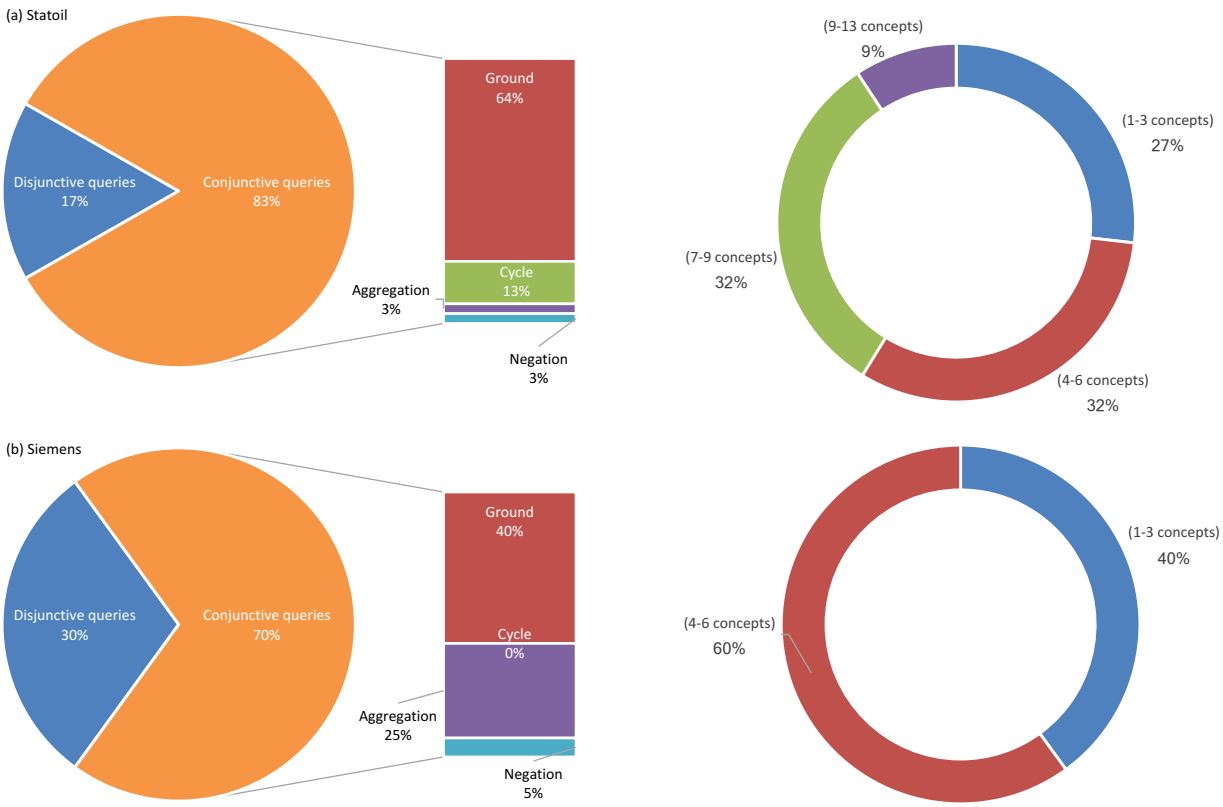

Fig. 2. An analysis of the Statoil and Siemens query catalogues.

relational databases reveals a substantial number of findings [21]. We employed the framework suggested by Catarci et al. [21] and considered dimensions presented in Table 2 to identify suggested paradigms for each dimension in the order of priority. All queries in the query catalogues are unique and cover a wide range of typical information needs. The query catalogues show that 73% respectively 60% of queries in-

Table 2
Framework for selecting the representation paradigms.

Dimension	Level	Support	Suggested paradigms
Frequency of interaction	Frequent	Uses cases (E4, E8)	1. form-based and 2. diagram-based
Variance of query tasks	Extemporary	Use cases (E3, E6) and query catalogues	1. icon-based and 2. diagram-based
Structural complexity	Sophisticated	Use cases (E2, E7) and query catalogues	1. diagram-based
Domain familiarity	Familiar	Use cases (E1, E5)	1. diagram-based

volve more than three concepts, referring to a high structural complexity (Figure 2). These evaluations led us to the second requirement:

(R2) Provide a multi-paradigm user-interface where a diagram-based paradigm has the central role and is supported by form-based and iconic representation paradigms.

This requirement is inline with one global finding that visual query tools that combine multiple representation and interaction paradigms are better to address varying user, task, and query types [21,46].

One should note that the Siemens case focuses on streaming sensor data (i.e., temporal queries), which leads to somewhat more domain-specific requirements on the user interface – i.e., the possibility to involve stream properties and to select relevant stream templates and parameters. This is also partly valid for Statoil as queries often deal with geographical data (i.e., spatial queries), and domain experts would benefit a lot from a map component for constraining and selecting data values. Therefore, a third requirement also needs to be met:

(R3) Provide domain-specific components for dealing with temporal and spatial data sources.

5.2. *Quality attributes*

We derived a set of quality attributes for VQSs from an end-user development perspective in order to meet C3. Quality attributes are non-functional requirements that effect run-time behaviour, design, and user experience and effectively increase the benefits gained and decrease the cost of adoption for end users [96]. We followed the approach employed by Khalili and Auer [48] and extracted a set of quality attributes. For this purpose, we used the conceptual survey we conducted earlier [80,91] as well as input we received from the use case partners. In the following, we describe the attributes, which are relevant in our context.

(A1) **Usability** refers to the capacity of a system to meet its identified aims and is measured in terms of its effectiveness (i.e., accuracy and completeness), efficiency (i.e., time/effort required), learnability (i.e., time and effort required to learn the tool), and user satisfaction.

(A2) **Modularity** refers to the degree which a system's components are independent and interlocking. A highly modular system ensures flexibility and extensibility, so that new components can easily be introduced to adapt to changing requirements and to extend and enrich the functionality provided.

(A3) **Scalability** in our context refers to the ability of a VQS to visualise and deal with large ontologies. A scalable VQS increases comprehensibility and reduces cognitive load by avoiding the cluttering and scattering of presentation, which in turn makes formulation and exploration easier against large ontologies.

(A4) **Adaptivity** refers to the ability of a system to alter its behaviour, presentation, and content with respect to context. A VQS could reduce the effort required for query formulation by adaptively offering concepts and properties, for instance with respect to previously executed queries.

(A5) **Adaptability**, in contrast to adaptivity, is a manual process whereby users customise a system to their own needs and contexts. An adaptable VQS could provide flexibility against changing requirements, e.g., one can add a new domain-specific representation component.

(A6) **Extensibility** refers to the ability and the degree of effort required to extend a system. An extensible VQS provides flexibility against changing requirements by providing room, from both architectural and design perspectives, for sustainable evolution.

(A7) **Interoperability** refers to the ability of a system to communicate and exchange data with other applications. Interoperability contributes to the functionality of a VQS by allowing it to utilise or feed other applications in an organisational workflow or digital ecosystem.

(A8) **Portability** refers to the ability of a VQS to query other domains, rather than only a specific domain, without high installation and configuration costs. Domain-specific components (e.g., presentation modules) could be offered if available; however, the lack of domain-specific components should not be blocking.

(A9) **Reusability** in our context refers to ability of a VQS to utilise queries as consumable resources. Reusability could decrease the learning effort by utilising previous queries for didactic purposes, and could also allow users to formulate more complex queries by modifying existing ones.

5.3. Discussion

A VQS should not be considered in isolation from the *context*, which could be characterised by a variety of dimensions such as user, task, data, and organisation [27,81]. In this respect, quality attributes presented previously are related and support usability directly/indirectly. They mainly ensure sustainability against potential variances in context. In other words, they support the evolution of a VQS against ever-changing context dimensions without losing the expressiveness-usability balance. For example, the heterogeneity of data necessitates domain-specific presentation and interaction components for improved user experiences. In this respect, modularity and extensibility plays an underpinning role by facilitating the development and integration of such components. Another example would be the organisational context: a VQS is often a part of larger tool portfolio for data extraction, analysis, and decision-making, and in this context interoperability is valuable to ensure a seamless orchestration.

One of the main problems that typical VQSs face is the scalability against large ontologies [46]. A VQS has to provide its users with fragments of the ontology (e.g., concepts and properties) continuously, so that users can select relevant ontology elements and iteratively construct their queries. However, even with considerably small ontologies, the number of concepts and properties to choose from increases drastically due to the propagation of property restrictions [25]. In turn, the high number of ontology elements overloads the user interface and hinders usability (i.e., scattering and cluttering). This can be approached with *progressive disclosure* technique meaning disclosing only the minimal amount of information and functionality required for the task at hand gradually and on demand. Another prominent approach is adaptivity [16], that is, in our context selecting and displaying the most relevant fragments of the ontology at each step.

The structural complexity of query tasks deserves special attention for choosing the right representation and interaction paradigms. Our use cases come with non-simple query tasks, which are structurally complex. Respectively, navigational interaction style becomes essential, i.e., *query by navigation* (QbN) [83,97]. Recent faceted search approaches, which are originally used to browse instances of a single concept, strive to offer the possibility to navigate and combine a number of concepts and create complex structures to retrieve data (e.g., [6,15]). We consider graph-based representation and navigation as an appropriate choice in this respect. This is because graphs are effective mechanisms to navigate, construct, and communicate complex topological structures for end users [21,46]. Secondly, it is well-known that the majority of end-user queries are conjunctive, and thus, in the semantic web setting, they could naturally be seen as graphs since we are dealing with unary and binary predicates only.

6. OptiqueVQS

OptiqueVQS is composed of an interface and a navigation graph extracted from the underlying ontologies. The interface components are populated and driven according to the information in the navigation graph. In the following subsections, we present each part.

OptiqueVQS generates non-temporal queries in SPARQL [40], while in temporal cases generates queries in STARQL [66]. STARQL provides an expressive declarative interface to both historical and streaming data. We chose STARQL since it supports OBDA, but OptiqueVQS could potentially generate queries in any other language. Technical details of STARQL are beyond the scope of this article; interested readers are referred to relevant material [57,66,92].

Regarding the spatial queries, OptiqueVQS generates queries still in SPARQL and qualitative spatial predicates (containment, overlap etc.) are supported by VQS as regular predicates. Their special meaning is taken care of by the OBDA platform, either by mapping to geospatial operations in the database, or to some materialised representation of these relations [56]. Besides, a map component allows selecting enti-

ties based on their geospatial location, rather than by name.

OptiqueVQS is available online together with the whole Optique platform, a comprehensive tutorial, and an example OBDA scenario including an ontology, a sample data set, and mappings for online testing and download[4].

6.1. OptiqueVQS frontend

The OptiqueVQS interface is designed as a *widget-based user-interface mashup* (UI mashup), which aggregates a set of applications in the form of *widgets* in a common graphical space and orchestrates them for achieving common goals [84]. Apart from flexibility and extensibility, such a modular approach provides us with the ability to combine multiple representations and interaction paradigms, and distribute functionality to appropriate widgets.

Initially, three widgets appear in OptiqueVQS, as depicted in Figure 3 (recall R2 at Section 5):

(W1) The first widget is a menu-based QbN widget accompanied with icons that allows the user to navigate concepts by picking relationships between them (see the bottom-left part of Figure 3).
(W2) The second widget is form-based and presents the attributes of a selected concept for selection and projection operations (see the bottom-right part of Figure 3).
(W3) The third widget is diagram-based and presents the constructed query and affordances for manipulation (see the top part of Figure 3) .

On the one hand, W1 and W2 provide a *view*; i.e., they focus the user to the current phase of the task at hand by providing means for gradual and on-demand exploration and construction. On the other hand, W3 provides an *overview*, i.e., an outlook of the query formulated so far, and lets the user refocus. These three widgets are orchestrated by the system, through harvesting event notifications generated by each widget as the user interacts.

A typical interaction between the user and the interface happens as follows:

1. the user first selects a *kernel* concept, i.e., the starting concept, from W1, which initially lists all domain concepts with their descriptions;
2. the selected concept appears on the graph (i.e., W3) as a *variable node* and becomes the *pivot* (*active*, *focus*) node (i.e., the node coloured in orange or highlighted);
3. W2 displays the attributes of the selected variable node in the form of text fields, range sliders, etc., so that the user can select them for output or constrain them;
4. the attributes selected for output (i.e., using the "eye" button) appear on the corresponding variable node with a letter "o", while constrained attributes appear with a letter "c";
5. the user can further refine the type of variable node from W2 by selecting appropriate subclasses, which are treated as a special attribute (named "Type") and presented as a multi-selection combo-box form element;
6. once there is a pivot node, each item in W1 represents a combination of a possible relationship-range concept pair pertaining to the pivot (i.e., indeed a path of length one);
7. a selection of path/item in W1 triggers a join between the pivot and the new variable node (of type range concept) over the specified relationship, and the new variable node becomes the focus (i.e., *pivoting*).

The user has to follow the same steps to involve new concepts in the query and can always jump to a specific part of the query by clicking on the corresponding variable node in W3. The arcs that connect variable nodes do not have any direction, but are implicitly read left to right. This is because for each active node only outgoing relationships and inverses of incoming relationships are presented for selection in W1. An example query is depicted in Figure 3 for the Statoil use case. The query asks for all the wellbores that belong to a development well and are operated by a company. In the output, we want to see the name of the wellbore, the synchronisation date and the name of the company.

The user can delete nodes, access the query catalogue, save/load queries and undo/redo actions through affordances provided by the buttons at the bottom part of W3. W3 indeed acts as a master widget, since it possesses the whole query and deals with its persistence. The user can re-use existing queries stored in the system by anyone, hence could modify an existing query to fit his/her current needs.

The user can also switch to an editable SPARQL mode and see the textual form of a query by clicking on "SPARQL Query" button at the bottom-right part of

[4] Access to OptiqueVQS online demo and the whole Optique platform with an example OBDA scenario: `http://sws.ifi.uio.no/project/optique-vqs/`

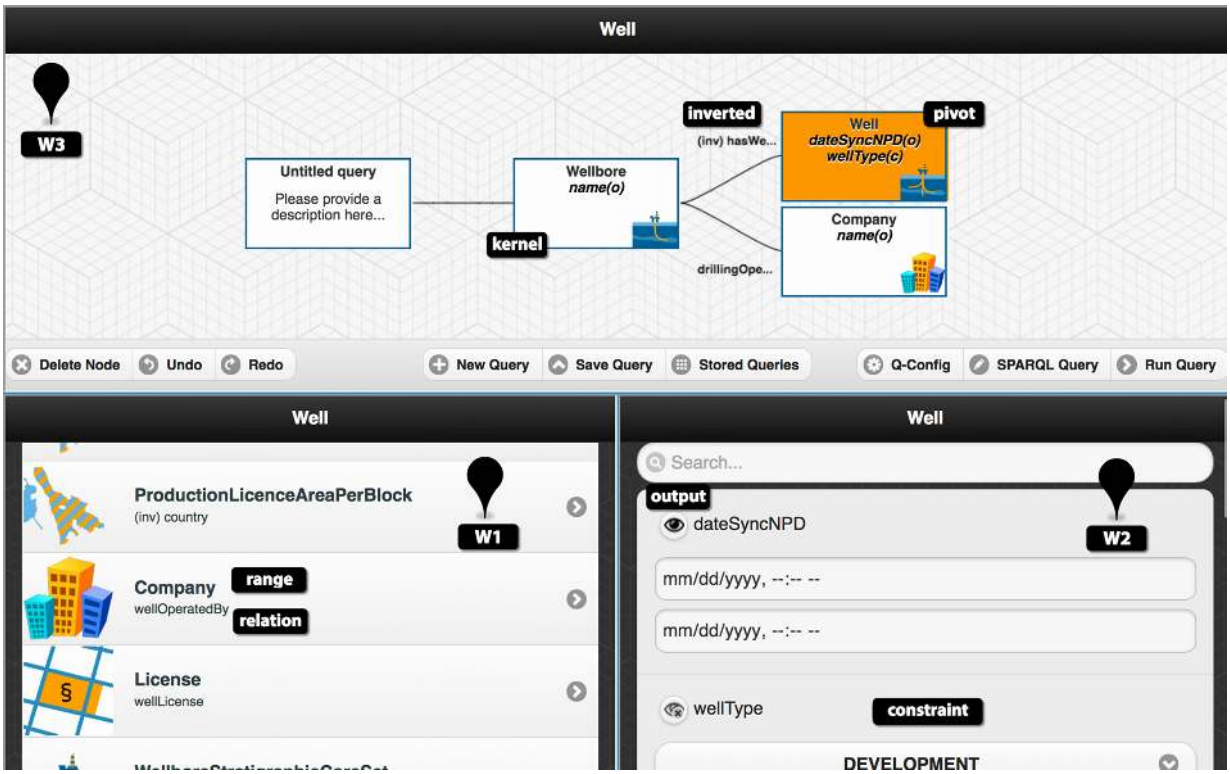

Fig. 3. OptiqueVQS interface – an example query in visual mode.

W3 as depicted in Figure 4. The user can keep interacting with the system in the textual form and continue to the formulation process by interacting with the widgets. For this purpose, the pivot/focus variable node text is highlighted and every variable node text is associated with a hyperlink to allow users to change the focus. Availability of the textual mode and its synchronisation with the visual mode enable us to realise collaboration between end users and IT experts. Especially for highly complex queries, IT experts could provide help on the textual mode, which they are expected to be more comfortable with, while end users can keep working on the visual mode. Moreover, from a didactic perspective, end users, who are eager to learn the textual query language, could switch between two modes and see the new query fragments being added/deleted after each interaction. Note that the SPARQL mode is constrained, in terms of expressiveness, to what can be represented in the visual mode.

We extended OptiqueVQS with three new widgets, which provide evidence on how a widget-based architecture allows us to distribute and hide complex functionality to/behind layers and combine different paradigms. One widget is for viewing example results, the other two widgets are addressing spatial and temporal use cases. They are activated by annotating (via OWL annotations) relevant properties as temporal or spatial (recall R3 in Section 5). The widgets are described as follows:

(W4) The fourth widget is a tabular result widget and appears as soon as the user clicks on the "Run Query" button (see Figure 5). It provides an example result list for the current query and also affordances for aggregation and sequencing operations.

Aggregation and sequencing operations fit naturally to a tabular view, since it is a related and familiar metaphor. Users can also view the full result list, inspect the individuals, and export data. For these purposes, in Optique, we use the Information Workbench (IWB)[5] [39,49], which is a generic platform for semantic data management.

[5] http://www.fluidops.com/en/products/information_workbench/

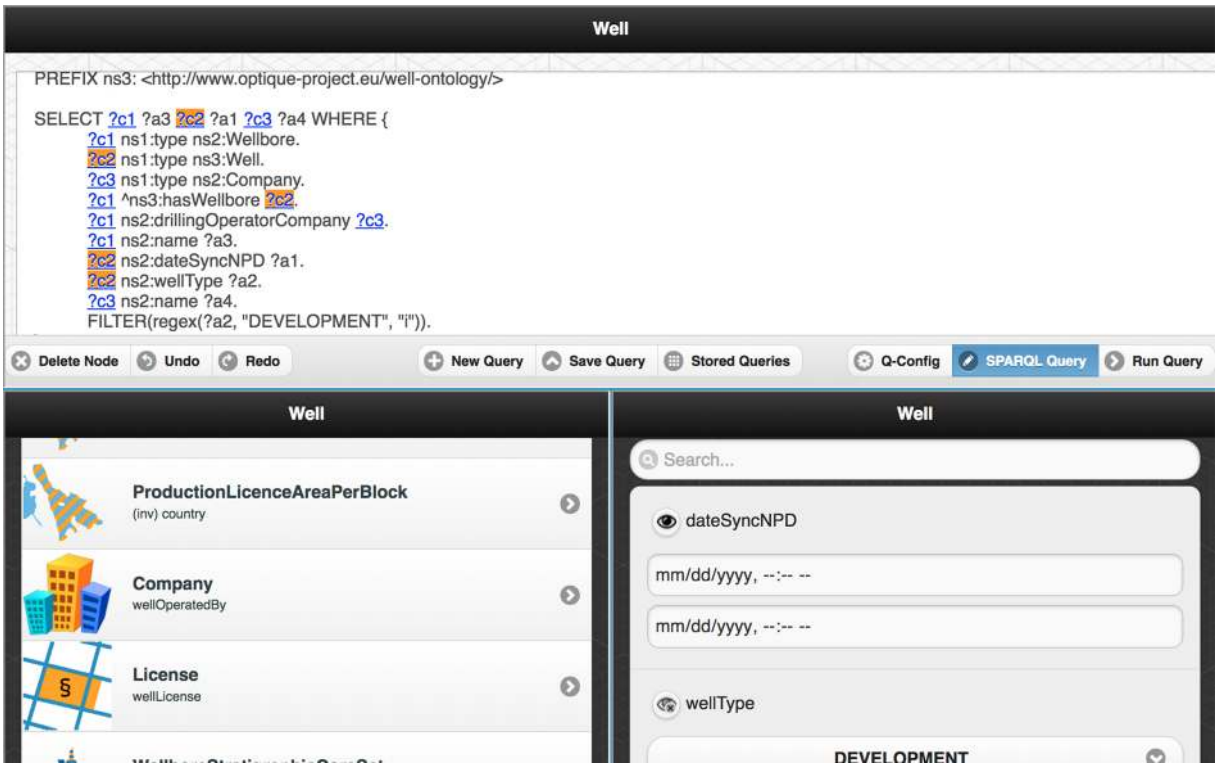

Fig. 4. OptiqueVQS interface – an example query in textual mode.

(W5) The fifth widget is a map widget. It is a domain-specific component for Statoil use case, and it allows end users to constrain attributes by selecting an input value from the map (see Figure 6).

A button with a pin icon is placed next to every appropriate (i.e., annotated as spatial) attribute presented in W2 to activate the map widget.

(W6) The sixth widget is a domain-specific component and supports temporal queries in the context of Siemens use case (see Figure 7).

OptiqueVQS produces temporal queries in STARQL. OptiqueVQS switches to STARQL mode when the user selects a dynamic property (i.e., whose extensions are time dependent, and coloured in blue). A stream button appears on top of W1 and lets the user configure streaming parameters such as slide (i.e., frequency at which the window content is updated/moves forward) and window width interval. If the user clicks on the "Run Query" button, a template selection widget (W4) appears for selecting a template for each stream attribute, which is by default "echo" (see Figure 8); W4 is normally used for displaying example results in SPARQL mode. The example query depicted in Figure 7 and Figure 8 asks for a train with turbine named "Bearing Assembly", and queries for the journal bearing temperature reading in the generator. The user can register the query in W4 by clicking on the "Register query" button for continuous execution.

6.1.1. Design rationale

The usability of OptiqueVQS is built on several design choices. In this section, we address the local design choices concerning the implementation of individual widgets. Major local design choices involve:

(a) *tree-shaped query representation* (W3) is meant to increase comprehensibility compared to generic graph representations with arcs and nodes directed and placed to arbitrary points;
(b) *inverted object properties* (W1 and W3) ensure a direction-free query representation and navigation in order to increase the visual readability of the query formulated;
(c) *object property – range concept pairs* (W1) decrease the number of navigational steps; i.e., rather than selecting an object property then a

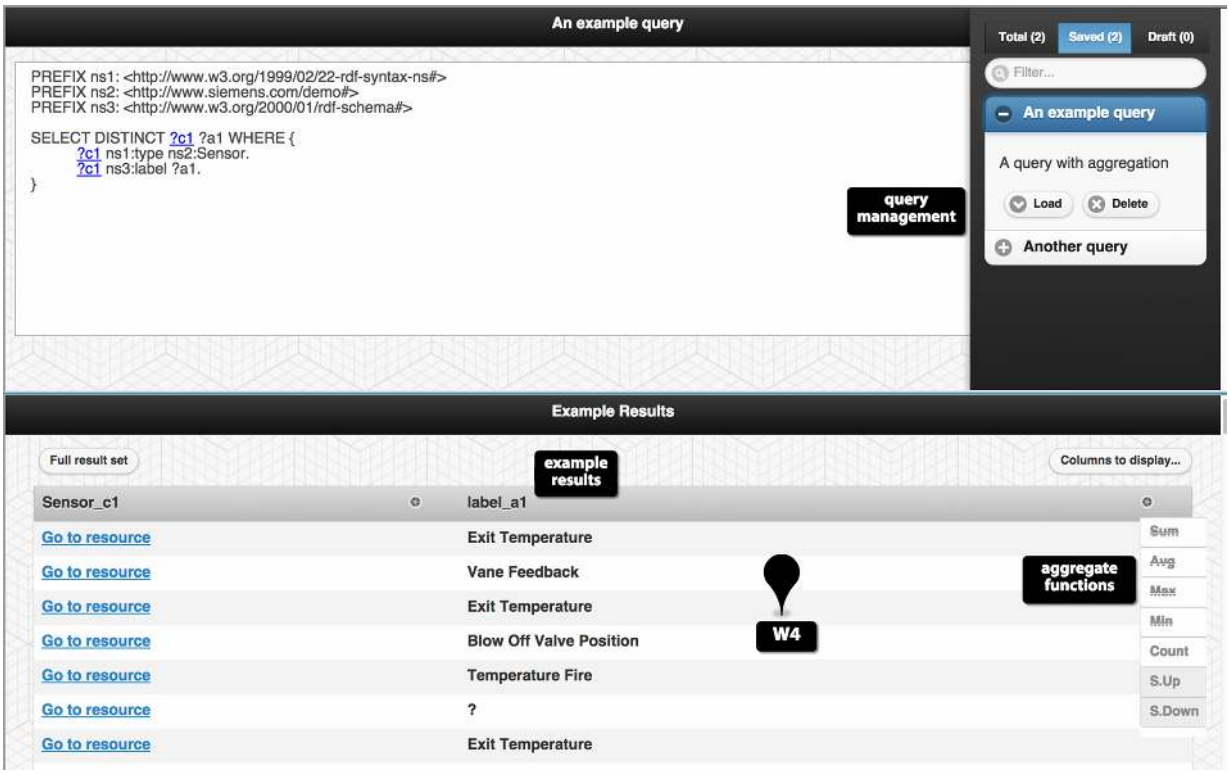

Fig. 5. OptiqueVQS interface – the tabular result widget with aggregation and sequencing support.

range concept, the user can select a pair at a single step;

(d) *simplified type refinement* (W2) reduces the type refinement to the attribute level; that is, the list of subclasses presented as an ordinary form element to provide a simplified and familiar solution.

In general, such design choices provide an orderly presentation and hide complexity and technical jargon related to the graphs, query language, and ontologies effectively, so as to reduce the cognitive load and knowledge and skills required. The semantics and syntax of the underlying query language and ontology are not delivered as they are; however, a correct translation from end-user operations to the query language is ensured.

For example, in a variant of OptiqueVQS, a graph representation is employed along with ingoing/outgoing arc distinction. In a user study with casual users, the participants complained about disorder in the presentation and their confusion due to the ingoing/outgoing relation distinction [103]. However, in another study with original OptiqueVQS with casual users, the participants praised the order and simplicity of the tree-shaped presentation [89].

6.2. *OptiqueVQS backend*

In this section, we present the main components of the OptiqueVQS backend. Currently, the OptiqueVQS backend relies on the infrastructure provided by the IWB. The IWB provides the OptiqueVQS backend with a triple store for storing ontologies, query logs, (excerpts of) query answers, etc., and generic interfaces and APIs for semantic data management (e.g. an ontology processing API). We have also started the implementation of a standalone version of the OptiqueVQS, which will not rely on the IWB.[6] In Figure 9, one can see the main components within the OptiqueVQS backend.

The frontend communicates with the backend via a REST API that returns a JSON object according to the performed request. The backend is in charge of

[6]For updates, see `http://sws.ifi.uio.no/project/optique-vqs/`

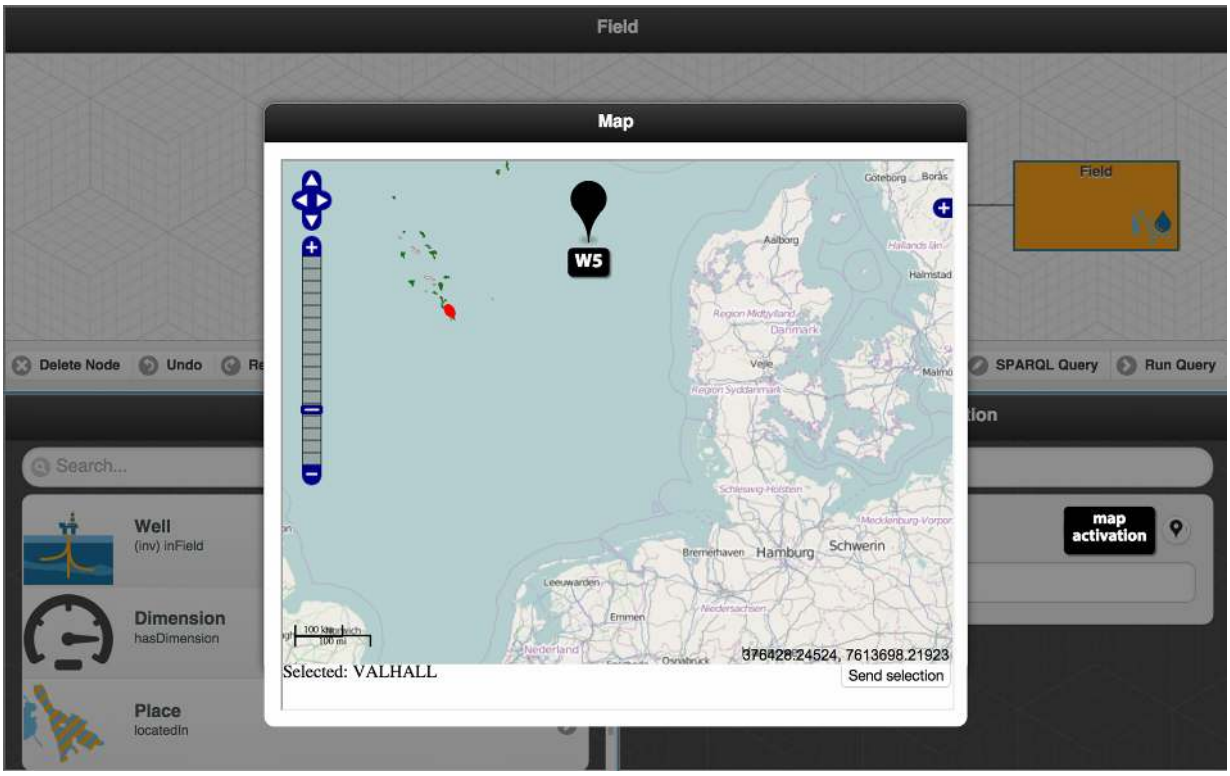

Fig. 6. OptiqueVQS interface – the map widget.

accessing *(i)* the ontology, which drives the information displayed in the frontend, and *(ii)* the query log, which plays an important role in ranking [87] as well as serving examples for the formulation of similar future queries.

The ontology can optionally be enriched with additional axioms to capture values that are frequently used and rarely changed (refer to the *data sampler* in the architecture); this includes the list of values and numerical ranges in an OWL data property range (i.e., for max/min sliders and drop-down boxes in W2). We use OWL annotation properties for this purpose.

The number of suggestions presented in W1 and W2 may grow quickly due to ontology size, number of relationships between concepts, inverse properties, and the propagative effect of inheritance of restrictions etc. As the lists grow, the time required for a user to find elements of interest increases; therefore, *adaptive query formulation*, i.e., ranking ontology elements with respect to previously executed queries (i.e. a query log), is a critical aspect in OptiqueVQS (refer to the *ranking component*). We implemented a light version of the ranking method described in Soylu et al. [87]. OptiqueVQS ranks suggestions presented in W1 and W2 with respect to the *partial query* that the user has constructed so far and the query history (i.e., context-aware). A given partial query is compared against similar queries in the query log and a rank is calculated for each possible extension accordingly.

The main component of the backend is the *graph projector* (described in the next section), which creates a navigation graph according to the ontology axioms. The graph projector in conjunction with the *VQS feeder* drives the population of the frontend widgets. Regarding the synchronisation of the widgets and the underlying graph, in its initial status OptiqueVQS lists all ontology concepts in W1, while W2 and W3 are empty. Hence, when initialising the OptiqueVQS frontend the backend returns a JSON object containing a list of all the concepts in the ontology. When a concept in W1 (or W3) is selected, that concept becomes the pivot or focus. This selection (from the frontend) triggers three requests (associated to the pivot concept) to the backend, which returns a JSON object for each of the following:

– neighbour concepts of the pivot to populate W1;

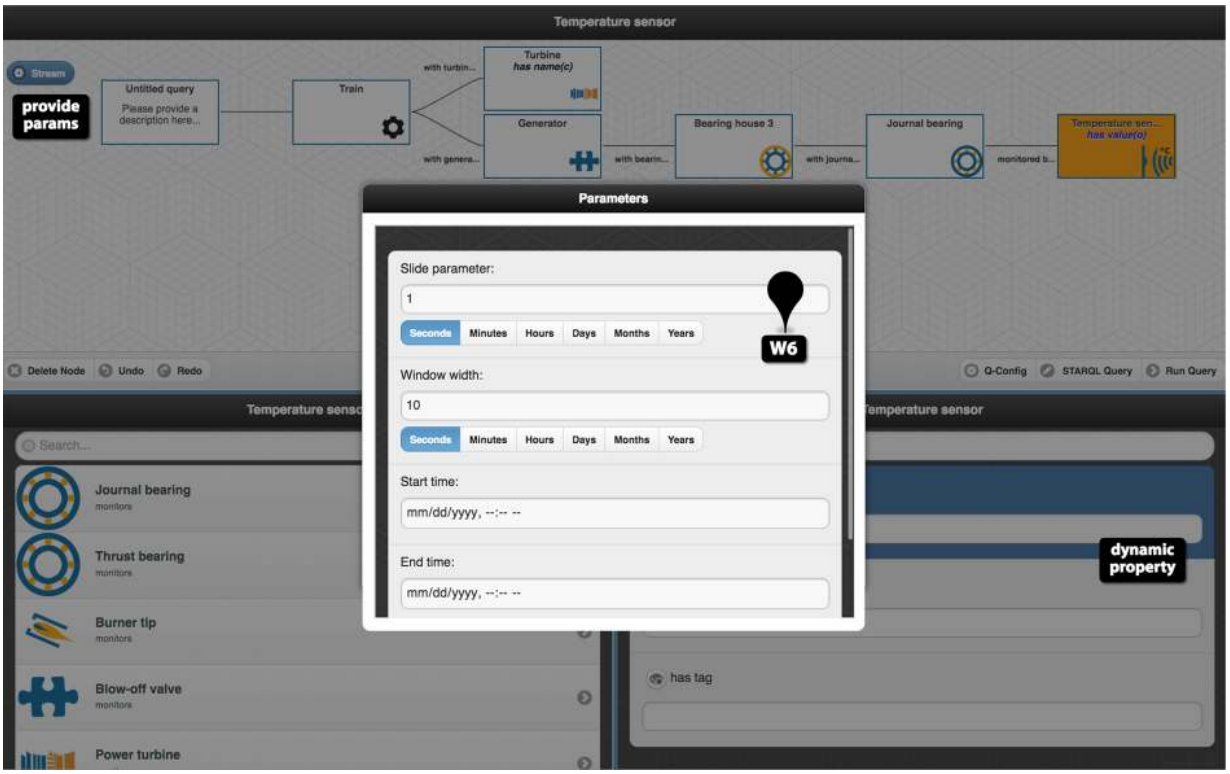

Fig. 7. OptiqueVQS interface – the stream parameter selection widget.

– attributes of the pivot to populate W2;
– and subclasses of the pivot to populate attribute "Type" in W2.

Regarding the backend scalability, firstly the computation of the navigation graph as well as enhancement with annotations necessary is done offline; therefore, OptiqueVQS is not doing any heavy computations such as reasoning in real time. Note that the size of the navigation graph is primarily determined by the size of the *terminological knowledge* (T-box), which is typically much smaller than the size of the *assertional knowledge* (A-box), i.e., data. Secondly, the navigation graph is kept in the memory on the backend to efficiently serve requests from the OptiqueVQS frontend and the synchronisation between the frontend and the backend, as described, is gradual and on demand, that is, every time only a small fragment of the ontology is requested and returned. Nevertheless, if a very large ontology is to be used in OptiqueVQS (e.g., SNOMED CT [93]), *ontology modularisation* techniques (e.g., [23,75]) might also be integrated into OptiqueVQS to enhance the user experience by extracting the relevant ontology module. For example, the entities in the query log for a specific user or user group could be used as a seed signature for the extraction of a relevant ontology module.

6.2.1. Ontology-driven navigation graph

From our work on the use cases, we discovered that end users ask mostly schema-level queries, e.g., "give me all wellbores that are located in a certain area". Thus, we are targeting at query formulation that is done in terms of classes and properties. OWL 2 axioms, on the other hand, can be exploited to help a user in navigating between classes and properties. For example, if a user during query formulation has the concept Wellbore active, then a query formulation system could suggest them to connect Wellbore with Core via hasCore due to the axiom '*Wellbore* SubClassOf: *hasCore* some *Core*'. Moreover, most of users' queries have a graph-like structure, where nodes are labelled with concepts and edges with properties. However, OWL 2 axioms are not well-suited for a graph-based navigation. Indeed, note that OWL 2 axioms do not have a natural correspondence to a graph, e.g., an OWL 2 axiom of the form 'C_1 and C_2 SubClassOf: D_1 or D_2' can be

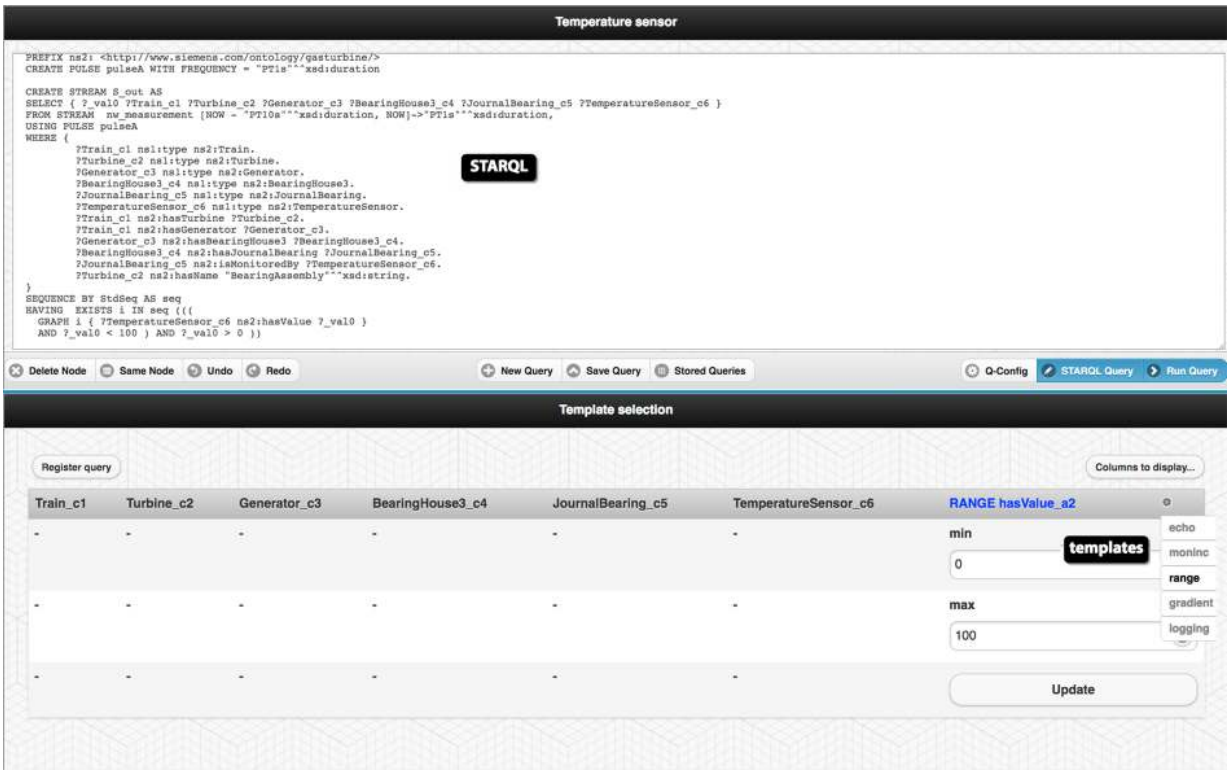

Fig. 8. OptiqueVQS interface – template selection for a stream query.

hardly seen as a graph. Even in the case when an axiom can naturally be seen as a graph, to the best of our knowledge there is no standard means to translate it to a graph. Therefore, we need a technique to extract a suitable graph-like structure from a set of OWL 2 axioms. To this end, we adapted a technique called *navigation graph* [6,7].

The nodes of a navigation graph are unary predicates, constants (named individuals, literal values) or datatypes, and edges are labelled with possible relations between such elements, that is, binary predicates. The key property of a navigation graph is that every X-labelled edge (v, w) is justified by one or more axioms entailed by $\mathcal{O}$ which "semantically relates" v to w via X.

Definition 6.1. *Let $\mathcal{O}$ be an OWL 2 ontology. A* navigation graph *for $\mathcal{O}$ is a directed labelled multigraph G having as nodes unary predicates, constants or datatypes from $\mathcal{O}$ and s.t. each edge is labelled with a binary predicate from $\mathcal{O}$. Each edge e is justified by one or more axioms α_e s.t. $\mathcal{O} \models \alpha_e$ and α_e is of the form given next, where b is a named individual, l_i is a literal value, $A, A_{sup}, A_{sub}, B, B_i$ classes or unary predicates, R_o, R_o^- object properties, R_d a datatype property, dt a datatype (e.g. string, integer), and x, y numerical values:*
(i) Edges e of the form $A \xrightarrow{R_o} B$ are justified by the following OWL 2 axioms:

- 'A SubClassOf: R_o restriction B', where *restriction* is one of the following: *some* (existential restriction), *only* (universal restriction), *min* x (minimum cardinality), *max* x (maximum cardinality) and *exactly* x (exact cardinality). Note that axioms with an union of classes in the restriction (e.g. 'A SubClassOf: R restriction B_1 or ... or B_n') or an intersection of classes in the restriction (e.g. 'A SubClassOf: R restriction B_1 and ... and B_n') also justify edges of the form $A \xrightarrow{R_o} B_i$.
- A combination of range and domain axioms of the form: R_o Domain: A' and 'R_o Range: B'.
- 'A SubClassOf: R_o value b', and b being a member of the class B (e.g., 'b Types: B').
- 'R_o InverseOf: R_o^-' when the navigation graph includes the edge $B \xrightarrow{R_o^-} A$.

- Top-down propagation of restrictions: 'A SubClassOf: A_{sup}' when the navigation graph includes the edge $A_{sup} \xrightarrow{R_o} B$.
- Bottom-up propagation of restrictions: 'A_{sub} SubClassOf: A' when the navigation graph includes the edge $A_{sub} \xrightarrow{R_o} B$.

(ii) Edges e of the form $A \xrightarrow{R_d} dt$ are justified by the following OWL 2 axioms:

- 'A SubClassOf: R_d restriction dt', where *restriction* is one of the following: *some*, *only*, *min x*, *max x* and *exactly x*. Note that dt can be a OWL 2 built-in datatype or user-defined datatype which are typically expressed with a datatype restriction (e.g., 'A SubClassOf: R_d restriction $dt[> x, < y]$', where dt is restricted with the interval defined by x and y.)
- A combination of range and domain axioms of the form: 'R_d Domain: A' and 'R_d Range: dt' (or 'R_d Range: $dt[> x, < y]$').
- 'A SubClassOf: R_d value l', and l being a literal value of type dt.
- Top-down propagation of restrictions: 'A SubClassOf: A_{sup}' when the navigation graph includes the edge $A_{sup} \xrightarrow{R_d} dt$.
- Bottom-up propagation of restrictions: 'A_{sub} SubClassOf: A' when the navigation graph includes the edge $A_{sub} \xrightarrow{R_d} B$.

(iii) Edges e of the form $A \xrightarrow{R_d} l_i$ are justified by the following OWL 2 axioms:

- A SubClassOf: R_d restriction $\{l_1 \dots l_n\}$, where *restriction* is one of the following: *some*, *only*, *min x*, *max x* and *exactly x*; and $l_1 \dots l_n$ is an enumeration of literal values (typically of type 'string').
- A combination of range and domain axioms of the form: 'R_d Domain: A' and 'R_d Range: $l_1 \dots l_n\}$'.
- 'A SubClassOf: R_d value l_i'.

(iv) Edges e of the form $A \xrightarrow{broader} B$ are justified by the OWL 2 axiom: B SubClassOf: A .

The edges in the navigation graph are used to populate the frontend widgets (i.e. views) with suggestions to guide the end user in the formulation of the query. Edges of type (i) are used to populate W1, while edges of types (ii) and (iii) populate the attributes in W2 for the current focus concept A in W3. Edges of type (iii) and (ii) also guide the automatic customisation of W2 with specific input fields for a given datatype, pre-populated dropdown lists for enumeration of values (e.g., company names) and range sliders for datatype restrictions (e.g., min/max possible depth of wellbores). Edges of type (iv) populate the list of subclasses for the focus concept A, which are treated as the special attribute "Type" in W2. OptiqueVQS relies on the OWL 2 reasoner HermiT [33] to build the navigation graph (e.g., extraction of classification) in order to consider both explicit and implicit knowledge defined in the ontology O.

6.2.2. Query conformation to navigation graph

To realise the idea of ontology and data guided navigation, we require that interfaces *conform to* the navigation graph in the sense that the presence of every element on the interface is supported by a graph edge. In this way, we ensure that interfaces mimic the structure of (and implicit information in) the ontology and data and that the interface does not contain irrelevant (combinations of) elements.

Our goal is to help a user to construct such queries that would be "justified" by the navigation graph. We assume that all the definitions in this section are parametrised with a fixed ontology $\mathcal{O}$.

Definition 6.2. *Let Q be a conjunctive query. The* graph of *Q is the smallest multi-labelled directed graph G_Q with a node for each term in Q and a directed edge (x, y) for each atom $R(x, y)$ occurring in Q, where R is different from $\approx$. We say that Q is* tree-shaped *if G_Q is a tree. Moreover, a variable node x is labelled with a unary predicate A if the atom $A(x)$ occurs in Q, and an edge (t_1, t_2) is labelled with a binary predicate R if the atom $R(t_1, t_2)$ occurs in Q.*

Finally, we are ready to define the notion of conformation.

Definition 6.3. *Let Q be a conjunctive query and G a navigation graph. We say that Q* conforms to *G if for each edge (t_1, t_2) in the graph G_Q of Q the following holds:*

- *If t_1 and t_2 are variables, then for each label B of t_2 there is a label A of t_1 and a label R of (t_1, t_2) such that $A \xrightarrow{R} B$ is an edge in G.*
- *If t_1 is a variable and t_2 is a constant, then there is a label A of t_1 and a label R of (t_1, t_2) such that $A \xrightarrow{R} t_1$ is an edge in G.*

Now we describe the class of queries that can be generated using OptiqueVQS and show that they conform to the navigation graph underlying the system.

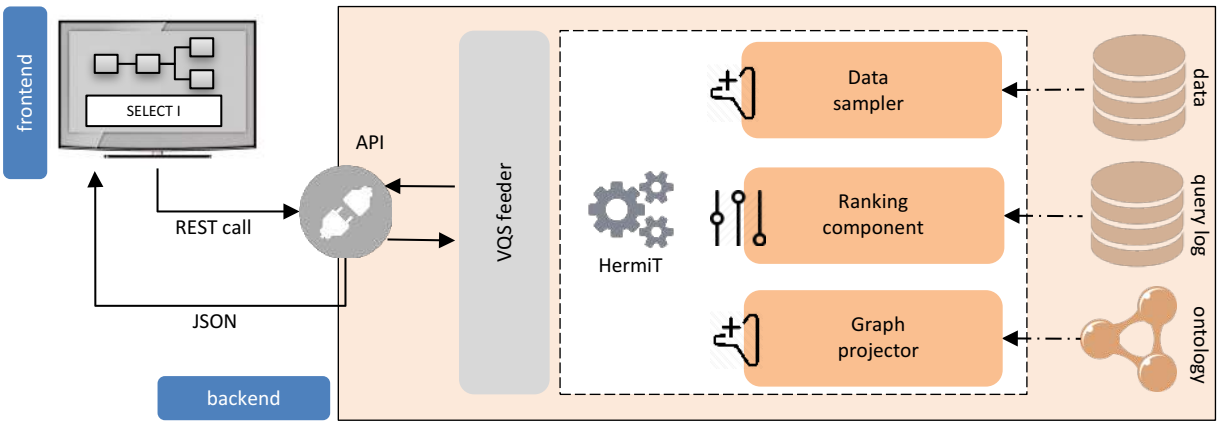

Fig. 9. OptiqueVQS backend architecture.

First, observe that the OptiqueVQS queries follow the following grammar:

$$\texttt{query} ::= A(x)(\wedge\, \texttt{constr}(x))^*(\wedge\, \texttt{expr}(x))^*,$$
$$\texttt{expr}(x) ::= \texttt{sug}(x,y)(\wedge\, \texttt{constr}(x))^*(\wedge\, \texttt{expr}(y))^*,$$
$$\texttt{constr}(x) ::= \exists y R(x,y) \mid R(x,y) \mid R(x,c),$$
$$\texttt{sug}(x,y) ::= Q(x,y) \wedge A(y),$$

where A is an atomic class, R is an atomic data property, Q is an object property, and c is a data value. The expression of the form $A(\wedge\, B)^*$ designates that B-expressions can appear in the formula $0, 1, \ldots$ times. An OptiqueVQS `query` is constructed using suggestions `sug` and constraints `constr` that are combined in expressions `expr`. Such queries are clearly conjunctive and tree-shaped (recall R1 in Section 5). All the variables that occur in classes and object properties are output variables and some variables occurring in data properties can also be output variables.

When users interact with OptiqueVQS,

- They start with a kernel class, as described above. Clearly, this initial query conforms to any navigation graph, including the one, underlying the system.
- Then, the system suggests the list of $\texttt{sug}(y,z)$ via W1 and of $\texttt{constr}(x)$ via W2 such that choosing any of them would leave the updated query conforming to the underlying navigation graph. In other words, all these choices are justified by the graph.

7. Quality Features

OptiqueVQS has the following interrelated features that are mapped to the quality attributes (i.e., A*n*) proposed at Section 5.2:

(F1) *View and overview* provide a continuous outlook of the query formulated so far while supplying the user with a set of possible actions. The goal is to ensure maximum end-user awareness and control [82] (A1).
Realisation: W3 provides a global overview of the user query, while W1 and W2 focus the user on the pivot for possible join, select, and projection operations.

(F2) *Exploration and construction* allow the user to navigate the conceptual space for exploration and construction purposes. Exploration could be also at instance level, in terms of cues (i.e., sample results) and instance level browsing [21,76] (A1).
Realisation: W1 and W2 suggest domain elements and allow ontology navigation. Each action adds reversible query fragments into the query. The user can also use the tabular result widget, i.e., W4, for example results.

(F3) *Collaborative query formulation* is meant to enable collaboration between users actively or passively. Such collaboration could be between an end user and an IT expert or between end users [63] (A1). Users can formulate more complex queries and improve their effectiveness and efficiency.
Realisation: OptiqueVQS synchronises visual and textual modes (i.e., *active collaboration* between IT experts and end users), allows users

to share queries (i.e., *passive collaboration*), and harnesses the query log to offer suggestions (i.e., passive).

(F4) *Query reuse* enables the user to reuse existing queries as they are or to modify them to construct more complex queries and/or to improve the effectiveness and efficiency (A1 and A9). Query reuse could indeed be considered a passive form of collaboration [63] (F3).
Realisation: OptiqueVQS allows users to store, load, and modify queries. Queries are stored in a query catalogue with descriptive texts to facilitate their search and retrieval.

(F5) *Spiral/layered design* (recall progressive disclosure) refers to distributing system functionality into layers [77], so as to enable an orderly access to the system, prevent complex functionalities to hinder the usability for less competent users (A1), view the ontology at different levels of detail (A3), tailor available functionality with respect to user needs (A4 and A5), and to add new functionalities without overloading the interface (A6).
Realisation: OptiqueVQS delegates functionality and ontology visualisation tasks to the different widgets. For instance, W4 offers aggregation and sequencing operations, while W2 presents data attributes and offers selection and projection functions.

(F6) *Gradual access* (recall progressive disclosure) is to cope with large ontologies with many concepts and properties. The amount of information that can be communicated on a finite display is limited. Therefore, gradual and on-demand access to the relevant parts of an ontology is necessary [46] (A1 and A3).
Realisation: W1 and W2 provide ontology elements adaptively and gradually on user demand, hence avoid cluttering and scattering the interface.

(F7) *Iterative formulation* allows the user to follow a formulate-inspect-reformulate cycle (A1), since a query is often not formulated in one iteration [63, 100].
Realisation: OptiqueVQS provides affordances to inspect, manipulate and extend a formulated query. For instance, users can freely change the pivot, delete nodes, and add new nodes from any point of the query.

(F8) *Ranked suggestions* improve the user efficiency by ranking ontology elements with respect to context, e.g., previous query log, and filtering down the amount of knowledge to be presented [87] (A1, A3, and A4). Ranking is a form of passive collaboration as it utilises queries formulated by others to provide gradual access (F3 and F6).
Realisation: OptiqueVQS offers a ranking method, which exploits the query history of users to rank and suggest ontology elements (in W1 and W2) with respect to a partial query that a user has constructed so far.

(F9) *Domain specific representations* support varied data types and domains. This ensures contextual delivery of data leading to immediate grasping [98] (A1). The availability of domain specific representations provides users and system with the opportunity to select representation paradigms that fit best to the data and task (A4 and A5).
Realisation: OptiqueVQS allows introducing new domain-specific widgets for visualisation and interaction, for instance, the map widget (W5) for geospatial interaction and visualisation.

(F10) *Multi-paradigm and multi-perspective* presentation is meant to combine multiple representation and interaction paradigms such as form and diagrams, and query formulation approaches such as visual query formulation and textual query editing, to meet diverse contexts [21,46] (A1). Moreover, the system and users can adapt the presentation (A4 and A5) and users can select among various paradigms depending on their role (F3), task (F2), and data at hand (F9).
Realisation: OptiqueVQS puts multiple representation and interaction paradigms (i.e., list/menus (W1), diagrams (W3), forms (W2), tables (W4)) as well as query formulation approaches (i.e., textual and visual) together.

(F11) *Modular architecture* allows new components to be easily introduced and combined in order to adapt to changing requirements and to support diverse user experiences (A1, A2 and A6). This could include alternative/complementary components for query formulation, exploration, visualisation, etc. with respect to context (A3, A4, A5, F9, and F10).
Realisation: OptiqueVQS is based on a UI mashup approach and is built on a widget-based architecture, where widgets are independent components acting as the building blocks. They communicate through broadcasting event notifications.

(F12) *Data exporting* enables the user to feed analytics tools with the data extracted for sense-making

processes, as they are not expected to have skills to transform data from one format to another. Therefore, means to export data in different format are required to ensure that the system fits into the organisational context (A7) and a broader user experience (A1).
Realisation: OptiqueVQS allows users to export data in various formats. For instance, in the context of Statoil use case, users can export query results in the format of their data analytics tools.

(F13) *Domain-agnostic backend* ensures domain independence. This allows a VQS to operate over different ontologies and datasets without any extensive manual customisation and code change [47,61] (A8).
Realisation: OptiqueVQS relies on a domain-agnostic backend. It projects the underlying ontology into a graph for exploration and query construction. Yet, it also allows domain-specific components to be introduced.

8. User Evaluation

The purpose of a VQS is to enable users to formulate queries effectively and efficiently. The *effectiveness* [13,20] is measured in terms of accuracy and completeness that users can achieve. The cost associated with the level of effectiveness achieved is called *efficiency* [13,20], and is mostly measured in terms of the time spent to complete a query. Note that, typically in *information retrieval* (IR), effectiveness is measured in terms of *precision*, *recall*, and *f-measure* (harmonic mean of precision and recall) over the result set; however, a VQS in our context is a *data retrieval* (DR) paradigm, for which a single missing or irrelevant object implies total failure [80]. In other words, data retrieval systems have no tolerance for missing or irrelevant results, while IR systems are variably insensitive to inaccuracies and errors, since they often interpret the original user query and the matching is assumed to indicate the likelihood of the relevance, rather than being exact [9,101]. Therefore, for a VQS, effectiveness is given in terms of a binary measure of success (i.e., correct/incorrect query) [47].

In the course of Optique project, we conducted a total of four industrial workshops with our use case partners (two for each use case). In the first set of workshops, we conducted unstructured interviews with domain experts and observed them in their daily routines. Shortly after the first set of workshops, we demonstrated a paper mock-up and had further discussions. A running prototype was developed iteratively with representative domain experts in the loop. At the second round of workshops, domain experts experimented with the prototype in a formal thinking-aloud session and we measured the effectiveness and efficiency of OptiqueVQS.

We also conducted two usability studies in a non-industrial context. The results are published elsewhere, but can be briefly summarised as follows:

(Ex1) *An experiment involving casual users* without any technical skills and knowledge. It was conducted on a generic domain. The results suggested that casual users without any technical background can effectively and efficiently use OptiqueVQS to formulate complex queries [89].

(Ex2) *A comparative experiment* comparing a variant of OptiqueVQS and a form-based query interface called PepeSearch [102]. The results suggested that OptiqueVQS is the preferred tool for formulating complex query tasks, while PepeSearch is the preferred tool for less experienced users for completing simple tasks [103].

In this article, we report the design and the results of the experiments that we conducted with our industrial partners:

(Ex3) *Statoil experiment* employed a bootstrapped (i.e., automatically generated [51,78]) oil and gas ontology[7] with 253 concepts, 208 relationships (including inverse properties), and 233 attributes [88].

(Ex4) *Siemens experiment without temporal queries* employed a manually constructed diagnostic ontology with five concepts, five relationships (excluding inverse properties), and nine attributes [88].

(Ex5) *Siemens experiment with temporal queries* employed a manually constructed turbine ontology with 40 concepts and 65 properties [90].

The ontologies, data, and information needs used in the experiments are provided by the industrial partners themselves and therefore are not artificial but reflect the reality and real interests.

[7] http://sws.ifi.uio.no/project/npd-v2/

Table 3
Profile information of the participants.

#	Age	Occupation	Exp.	Education	Tech. skills	Similar tools	Sem. Web
P1	39	Geologist	Ex3	Master	3	3	1
P2	40	Biostrat	Ex3	Master	2	1	1
P3	49	IT advisor	Ex3	Master	5	4	1
P4	33	Software engineer	Ex4	Bachelor	5	2	1
P5	27	Diagnostic Engineer	Ex4	Bachelor	5	5	1
P6	60	Mechanical Engineer	Ex4	Master	3	1	1
P7	45	Mechanical Engineer	Ex4	Bachelor	1	2	1
P8	37	R&D engineer	Ex5	PhD	4	1	1
P9	54	Diagnostics Engineer	Ex5	Bachelor	5	3	1
P10	39	Engineer	Ex5	PhD	5	2	1

8.1. Experiment design

The experiments were designed as a *thinking-aloud* study. Each participant performed the experiment in a single session, while being watched by an observer. Participants were instructed to think aloud, including any difficulties they encounter (e.g., frustration and confusion), while performing the given tasks. A five minutes introduction of the topic and tool was delivered to the participants along with an example before they were asked to fill in a profile survey. The survey asked users about their age, occupation and level of education, and asked them to rate their technical skills, such as on programming and query languages, and their familiarity with similar tools on a Likert scale (i.e., 1 for "not familiar at all," 5 for "very familiar"). Participants were then asked to formulate a set of information needs into queries with OptiqueVQS (i.e., tasks).

A number of empty queries, each corresponding to a task in the experiment, was generated in OptiqueVQS for each user. Users received their tasks one by one on paper, and for each task loaded the corresponding empty query. Formulating and executing a query, i.e., clicking "Run Query" button, and inspecting the result set constituted one attempt. Participants had a maximum of three attempts per task, this was enforced by the system (the "Run Query" button was blocked after three attempts). A task was ended when the participant indicated completion or exhausted his/her three attempts. Every attempt for each task was recorded by the OptiqueVQS as a draft query, along with the time taken for each attempt.

Three participants from Statoil and seven participants from Siemens took part in the experiments. The profiles of participants are summarised in Table 3, which shows that participants vary in technical skills and experience with similar tools and have no familiarity with semantic web technologies.

There were nine tasks for the Statoil experiment (Ex3), five tasks for the first Siemens experiment (Ex4), and five tasks for the second Siemens experiment with temporal queries (Ex5). Each task corresponds to a conjunctive query and is listed in Table 4. The key elements are highlighted in the context of this article for clarity.

8.2. Results

The results of all the three experiments are summarised in Figure 10 and Table 5.

Regarding the Statoil experiment (Ex3), a total of 27 tasks were completed by the participants. Results show 84% correct completion rate, 69% first-attempt correct completion rate (i.e., percentage of correctly formulated queries in the first attempt), and an average of 243 seconds and 1.4 attempts for completing a task. The first participant had only one incorrect, and the second participant had no incorrect task. T3 was about fields operated by Statoil, and the third participant formulated a Field - FieldOperator pair instead of a Field - Company pair. This confusion between FieldOperator and Company led him to incorrectly solve T5 as well. T7 not only takes the longest time but also the highest average attempts; participants raised that the ontology did not match their understanding of the domain and therefore they found it hard to formulate this query.

In the Siemens experiment without temporal queries (Ex4), a total of 18 tasks were completed by the participants. The third participant exceeded the allocated time for the session and could not attempt the last two tasks, therefore these are omitted from the results. Correct completion rate was 88%, while first-attempt correct completion rate was 72%. Average time and num-

Table 4
Information needs used in the experiments – marked tasks (*) are temporal .

#	Exp.	Information need
T1	Ex3	List all *fields*.
T2	Ex3	What is the water depth of the “Snorre A" *platform* (facility)?
T3	Ex3	List all *fields* operated by “Statoil Petroleum AS" *company*.
T4	Ex3	List all *exploration wellbores* with the *field* they belong to and the *geochronological era(s)* with which they are recorded.
T5	Ex3	List the *fields* that are currently operated by the *company* that operates the “Alta" *field*.
T6	Ex3	List the *companies* that are *licensees* in *production licenses* that own *fields* with a re-coverable oil equivalent over more than “300" in the *field reserve*.
T7	Ex3	List all *production licenses* that have a *field* with a *wellbore* completed between “1970" and “1980" and recoverable oil equivalent greater than “100" in the *company reserve*.
T8	Ex3	List the *blocks* that contain *wellbores* that are drilled by a *company* that is a *field operator*.
T9	Ex3	List all *producing fields* operated by “Statoil Petroleum AS" *company* that has a *wellbore* containing “gas" and a *wellbore* containing “oil".
T10	Ex4	Find all *assemblies* that exist in system.
T11	Ex4	Show all *messages* that *tribune* “NA0101/01" generated from “01.12.2009" to “02.12.2009".
T12	Ex4	Show all *turbines* that sent a *message* containing the text “Trip” between “01.12.2009" and “02.12.2009".
T13	Ex4	Show all *event* categories known to the system.
T14	Ex4	Show all *turbines* that sent a *message* category “Shutdown” between “01.12.2009" and “02.12.2009".
T15	Ex5	Display all *trains* that have a *turbine* and a *generator*.
T16	Ex5	Display all *turbines* together with the *temperature sensors* in their *burner tips*. Be sure to include the *turbine name* and the *burner tags*.
T17*	Ex5	For the *turbine named* “Bearing Assembly”, query for *temperature readings* of the *journal bearing* in the *compressor*. Display the reading as a simple echo.
T18*	Ex5	For a *train* with *turbine* named “Bearing Assembly”, query for the *journal bearing temperature* reading in the *generator*. Display readings as a simple echo.
T19*	Ex5	For the *turbine* named “Burner Assembly”, query for all *burner tip temperatures*. Display the readings if they increase monotonically.

Table 5
Aggregated experiment results: correct completion rate, first attempt correct completion rate, average time, and average number of attempts.

Exp.	CCR (%)	1stCCR (%)	Avg. time (s)	Avg. # of attempts
Ex3	84	69	243	1.4
Ex4	88	72	132	1.5
Ex5	100	66	103	1.3

ber of attempts for completing a task were 132 seconds and 1.5 respectively. The third and fourth participants had one incorrect task. Participants had a minor issue with the date format, therefore Task 11, where a date constraint appeared for the first time, took the longest time.

In the Siemens experiment with temporal queries (Ex5), a total of 15 tasks were completed by the participants. The results show 100% correct completion rate, 66% first-attempt correct completion rate, and an average of 103 seconds and 1.3 attempts for completing a task. Participants had a minor issue with the fact that they need click on the “Run Query” button in order to select a template from the tabular view – starting from the Task 17, which took the longest time. A straight forward solution for stream based queries would be to change the name of button to “Select a Template” to prevent confusion, as the “Run Query” button is originally meant for non-stream query tasks.

8.3. Discussion

Overall, the results indicate high effectiveness and efficiency suggesting that OptiqueVQS is a viable tool to visually construct considerably complex queries for querying structured data sources. All participants praised the utility of OptiqueVQS for formulating complex information needs into queries. A common statement was that such a solution will not only im-

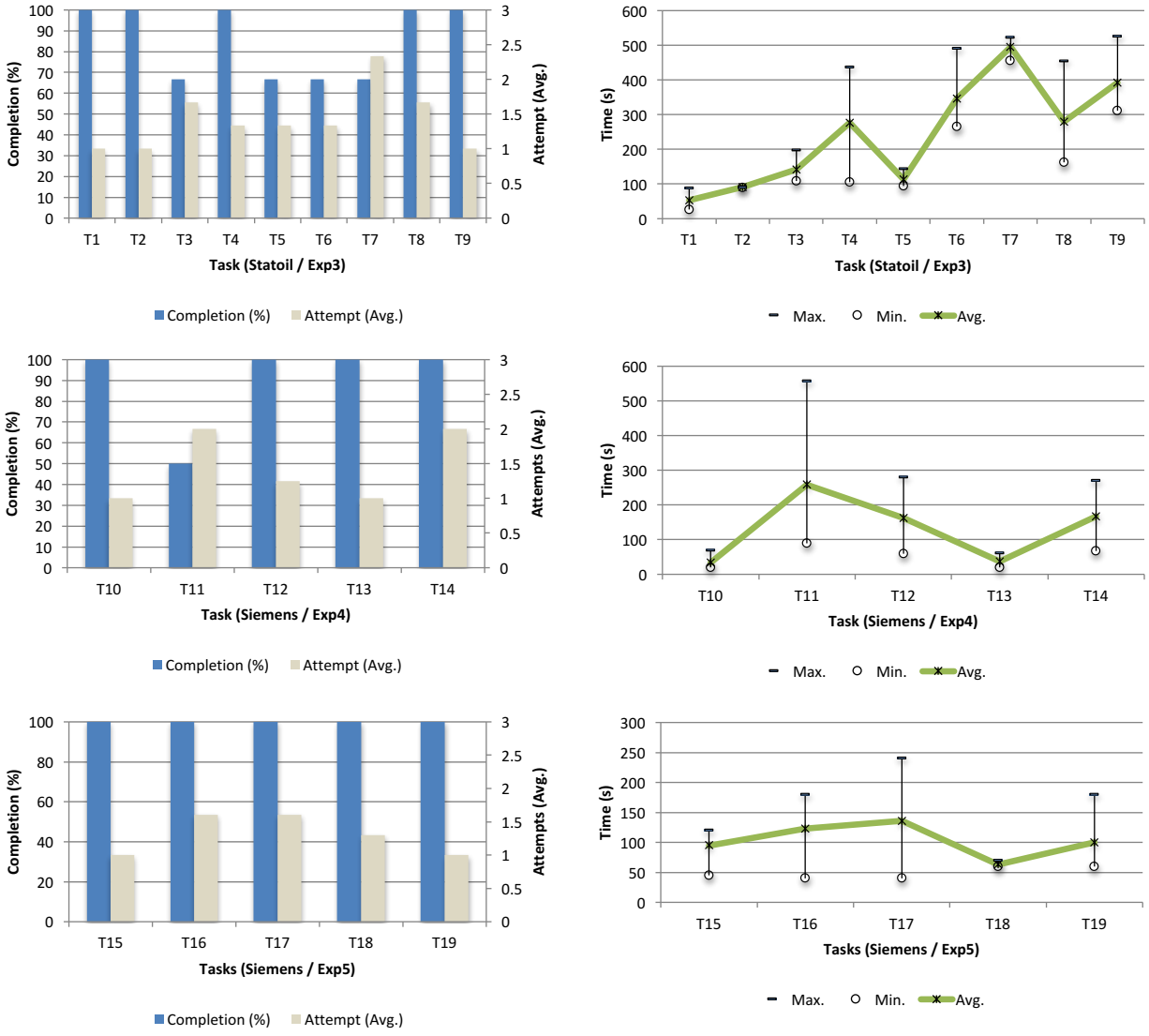

Fig. 10. Experiment results for the three usability studies with domain experts.

prove their current practices, but also augment their value creation potential due to the flexibility of formulating arbitrary queries. Three complex queries formulated by Statoil and Siemens domain experts and casual users (for reference) are given in Figure 11, Figure 12 and Figure 13 respectively. The first query was provided by a Statoil domain expert for the query catalogue and he estimated that he would need a full day to extract this information with the existing tools. On the other hand, the same Statoil user was able to formulate a query of similar complexity with OptiqueVQS within less than 10 minutes. The second query (Ex3) only took 63 seconds on average to complete by Siemens' domain experts. The third query took only 91 seconds to complete on average for a casual user [89].

If one compares the aggregated results presented in Table 5, it is noticeable that Statoil domain experts spent more time on average when completing a task, while the average number of attempts are similar. This has two reasons according to participants' feedback and our observations: *(i)* conceptual mismatch between the Statoil domain expert's understanding of domain and the ontology – note that the Statoil ontology was automatically bootstrapped with little manual fine tuning, while the Siemens diagnostic ontology

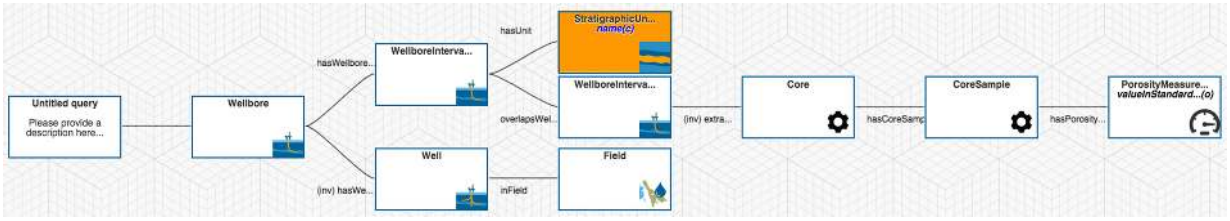

Fig. 11. A complex query formulated by Statoil's domain experts.

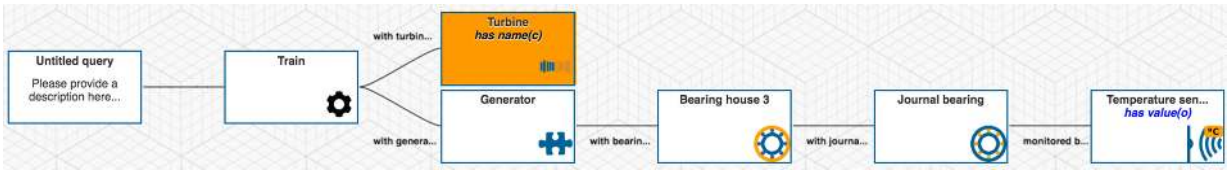

Fig. 12. A complex query formulated by Siemens' domain experts.

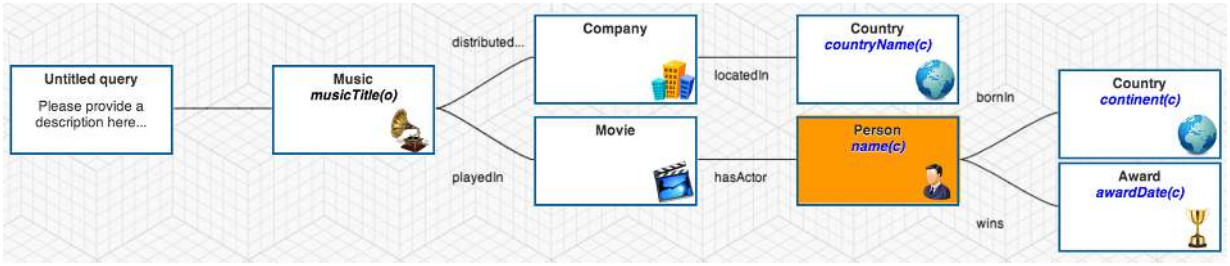

Fig. 13. A complex query formulated by casual users.

used in the experiment was smaller in size and was manually created (i.e., of higher quality); *(ii)* Statoil domain experts were more engaged with the data and spent considerably more time on checking the validity of the results, since the Statoil experiment was conducted on real data and participants were able to recognise and relate with the data, while in the Siemens case data was temporal and anonymised.

In general, participants raised two major issues. First, they asked for a longer training session; indeed, the training sessions were intentionally kept short in order to test the learnability of OptiqueVQS. The high completion rates, even with complex queries, suggest that the tool has high learnability. Secondly, participants pointed that the ontology did not always reflect their understanding of the domain, which was mostly an issue for the Statoil experiment. This issue is particularly likely to emerge in OBDA scenarios where an ontology is bootstrapped from database schemas and similar artefacts, since the ontology then will reflect the quality and nature of the given sources. Therefore, better heuristics are required for the automatic generation ontologies such as from database schemas as well as a careful manual fine tuning process. The usability of an ontology is as crucial as the usability of a query formulation tool; however, ontology usability is an overlooked issue in the research community and demands more attention.

Finally, a few participants had inquiries regarding the capabilities of OptiqueVQS:

- the ability to have advanced operations such as disjunction (asked for mainly by domain experts with IT skills and knowledge);
- the ability to combine and/or join concepts that are not directly linked; and
- automatic filtering of attributes and attribute values with respect to the previously selected constraints and the partial query at hand.

We will discuss these matters in Section 10.

9. Related Work

Visual approaches for querying structured semantic data sources are primarily categorised into VQLs and VQSs, as explained in Section 1. A VQS is built on an informal set of user actions that effectively capture a set of syntactic rules specifying a query language (e.g., [4,4,41]), while a VQL employs a formal visual notation and syntax corresponding to a textual query language (e.g., [11,38,43]). Such approaches can be further classified with respect to the main interaction paradigm.

Browsing (for query formulation purposes) and *schema navigation* are two prominent interaction paradigms. The former refers to interacting at an instance level, that is, the user browses the data set by adding and removing constraints and following the links between instances. Faceted search is a very good example of this paradigm, e.g., [6,15,99]. The latter is used by OptiqueVQS and refers to interacting at a conceptual level, that is, using an external vocabulary, for example provided by an ontology, to express the information need at the schema level, e.g., [4,37]. Browsing alone is often not good for meeting complex information needs and could be computationally problematic for very large data sets. The user is usually restricted to the individuals of a single concept and partial results need to be calculated for each possible future selection. Schema navigation is a better approach for meeting complex information needs even when both data and results sets are large.

Another categorisation arises from the source of vocabulary, which might be extracted from the data set or be provided by an external ontology. The former refers to extracting concepts and relationships by analysing the data set, i.e., extracting a pseudo ontology (e.g., [15,102]). This approach is adequate for cases where an ontology is not available, prevents the user from building unsatisfiable queries (i.e., no empty result sets), and allows using statistics about data for optimisation. The latter approach uses an ontology to feed the query formulation process, e.g., [95]. An ontology could be much more expressive than what one can extract from data, and the vocabulary extraction process could be quite expensive for large and dynamic data sets. For example, data sets in our use cases change very rapidly in vast amounts and this makes real time processing very hard. Offline processing is not an option as this would lead to missing and/or incorrect results; users need to access real time data. Finally, it is not always desirable to formulate queries with guaranteed results. For example, in the Siemens use case, most of the user queries specify an error situation in their hardware for which there is often no matching data at the time of query formulation (i.e., the user gets notified when the data changes and the query becomes satisfied). OptiqueVQS uses a hybrid approach, where an ontology is the main source of vocabulary and data set is used for a limited extent (recall Section 6.2).

Regarding the visual approaches in general, notable examples of VQLs are LUPOSDATE [36], RDF-GL [43], Nitelight [79], GQL [11], and QueryVOWL [38]. LUPOSDATE, RDF-GL and Nitelight follow RDF syntax at a very low level through node-link diagrams representing the subject-predicate-object notation, while GQL and QueryVOWL represent queries at comparatively higher level, such as with UML-based diagrams. Each of these languages are managed by a VQS providing means for construction and manipulation of queries in a visual form. Although VQL-based approaches with higher level of abstraction are closer to end users, the users still need to possess a higher level of knowledge and skills to understand the semantics of visual notation and syntax and to use it. Note that although OptiqueVQS uses a tree-shaped query representation, it is informal, simplified, and free of any syntax and jargon related to ontologies and query languages.

VQSs have a better potential to offer a good balance between expressiveness and usability. The prominent examples of VQSs are gFacet [41], OZONE [95], SparqlFilterFlow [37], Konduit VQB [4], Rhizomer [15], and PepeSearch [102]. gFacet, OZONE, and SparqlFilterFlow employ a diagram-based approach and diagrams representing the queries are rather informal. Konduit VQB and Rhizomer employ a form-based paradigm. Diagram-based approaches are good in providing a global overview; however, they remain insufficient alone for view (i.e., zooming into a specific concept for filtering and projection). This is because the visual space as a whole is mostly occupied with query overview. Form-based approaches provide a good view; however, they provide a poor overview, since the visual space as a whole is mostly occupied with the properties of the focus concept. Approaches combining multiple representation and interaction paradigms are known to be better since they can combine view and overview. gFacet and Rhizomer are originally meant for data browsing, that is they operate on data level rather than schema level and every user interaction generates and sends SPARQL queries in the background. Therefore, they are highly

Table 6

Comparison of related tools with respect to our industrial requirements (B = Browsing, S = Schema navigation, H = Hybrid, O = Ontology, D = Data; ✓= yes, Θ = partially, - = no).

Criteria/Tool	gFacet	OZONE	SparqlFilterFlow	Konduit VQB	Rhizomer	PepeSearch	Super Stream	TELIOS Spatial	OptiqueVQS
Interaction type	B	S	S	S	B	H	S	S	S
Vocabulary	D	O	D	D	D	D	D	O	H
Downloadable	✓	-	-	-	✓	✓	-	-	✓
Tree-shaped	✓	✓	✓	✓	✓	Θ	✓	✓	✓
Multi-paradigm	Θ	Θ	Θ	-	✓	-	-	✓	✓
Temporal	-	-	-	-	-	-	✓	-	Θ
Spatial	-	-	-	-	-	-	-	✓	Θ
Modularity	-	✓	-	-	-	-	✓	-	✓
Scalability	Θ	Θ	Θ	Θ	Θ	Θ	Θ	Θ	✓
Adaptivity	-	-	-	-	-	Θ	-	-	✓
Adaptability	-	✓	-	-	-	-	✓	-	✓
Extensibility	-	✓	-	-	-	-	✓	-	✓
Interoperability	-	-	-	-	-	-	-	Θ	Θ
Portability	✓	✓	✓	✓	✓	✓	✓	✓	✓
Reusability	-	✓	✓	Θ	-	-	✓	✓	✓

data-intensive, which is often impractical for large data sets. Finally, PepeSearch uses conventional forms and mixes schema-based search and browsing; search is limited with a kernel concept and concepts directly related to it, and relevant terms are extracted from the data. Apart from limited expressivity, PepeSearch suffers from poor domain knowledge extracted from data (compared to a rich ontology), although the interface is naturally tailored by the data. Also, it does not offer means to cope with large and frequently changing datasets (i.e., one needs to re-extract schema information if data changes).

As far as temporal queries are concerned, notable examples of temporal query languages in the Semantic Web are C-SPARQL [10], SPARQLstream [17], and CQELS [68]. These approaches extend SPARQL with a window operator whose content is a multi-set of variable bindings for the open variables in the query. However, in this paper we are rather interested in visual solutions sitting on top of any of these languages. Although several visual tools exist for SPARQL [91], the work is very limited for temporal languages. An example is SPARQL/CQELS visual editor designed for the Super Stream Collider framework [73]. However, the tool follows the jargon of the underlying language closely and is not appropriate for end users as it will demand considerable knowledge and skills.

Concerning spatial querying, notable formal textual query languages are stSPARQL [12] and geoSPARQL [67]. stSPARQL is an extension of SPARQL 1.1 for querying linked geospatial data that changes over time, while geoSPARQL is a recent standard of the Open Geospatial Consortium (OGC) for static geospatial data. Although there are numerous tools for visualising and interacting with spatial data such as Sextant [65], visual query tools are limited. A visual query tool, which is an adaptation from an earlier facet-based search tool, is being developed in the TELIOS project [28], by introducing a supplementary map component for constraining certain location dependent attributes.

Table 6 gives a comparison of the prominent tools. The summary suggests that none of the tools alone can address our industrial requirements. The majority of tools presented are either formal or have a strong focus on browsing, which leads them to be highly explorative and instance oriented. While browsing is very adequate for open Web, in our context, due to the large data size and the nature of the tasks, interacting with the ontology instead of directly with the data is more suitable for domain experts and computationally more feasible. OptiqueVQS is, however, a visual query sys-

tem working primarily at a conceptual level and it is not our concern to reflect the underlying formality (i.e., query language and ontology) per se. We are also not interested in providing full expressivity, as we aim to reach a usability-expressiveness balance. The design of OptiqueVQS is based on clear requirements, solid design choices with a rationale, and quality attributes. Finally, there is a lack of rigorous theoretical underpinning in the context of RDF and OWL 2. Existing approaches mostly either focus on RDF, thus essentially disregarding the role of OWL 2 ontologies, or do not reveal how underlying semantics are projected to drive exploration and query formulation.

10. Discussion

The use case analyses of Statoil and Siemens and experiments conducted with OptiqueVQS on different user groups (i.e., casual users and domain experts) and scenarios support two primary findings:

1. the majority of end user queries are tree-shaped and conjunctive [69]; and
2. a multi-paradigm design has good potential to support different types of users and tasks [21].

Currently OptiqueVQS supports 67% and 65% of the queries in the Statoil and Siemens query catalogues, respectively, which are tree-shaped conjunctive queries with aggregation and excluding negation (recall Figure 2). This is a considerable achievement towards supporting domain experts on meeting their own information needs. Domain experts had 90% average success rate overall in three experiments (recall Table 5) without any prior training, appreciated OptiqueVQS's design, and indeed suggested interesting improvements, which we will discuss below.

The overall process itself also revealed useful insights and blueprints towards a more systematic research and development of OptiqueVQS-like tools and systems. This includes topics such as analysis and categorisation of a representative query catalogue, query types, context of use, and potential representation and interaction paradigms. Design choices behind OptiqueVQS at macro and micro levels along with a list of quality attributes and interrelated supporting features are valuable as useful design practices. OptiqueVQS meets a good number of quality attributes compared to other existing systems thanks to this systematic approach (recall Table 6). That being said, in the following, we will discuss the limitations of OptiqueVQS and potential solutions.

Note that OptiqueVQS is designed to meet diverse user and task types, for a context of use involving frequent use, and varied and structurally complex query tasks. Therefore, a query interface with a specialised design and functionality, for example built on a single paradigm or having higher/lower level of expressiveness, could perform better for a different context. In our earlier comparative study [103], we found that casual users prefer a simpler form-based interface although they can successfully use a more advanced interface like OptiqueVQS. The reason is that when the tasks are structurally simple, it becomes a cognitive burden to use an interface with more functionality and extended design. Another example concerns querying temporal and spatial data sources. OptiqueVQS deals with such data in a non-specialised way due to the scenarios it primarily supports, that is, spatial and temporal aspects are not in the core of OptiqueVQS's design. For example, a specialised interface having a map representation in the core of its design could be required for data with high spatial orientation.

As discussed in Section 8, a few domain experts raised questions regarding the expressiveness level of OptiqueVQS and its capabilities. The first point concerns the formulation of more complex queries including such as disjunction. Expectedly, this was raised by domain experts with higher technical skills and knowledge. Implementing complex functionality is often not straightforward without compromising usability. Today, OptiqueVQS does not support this expressiveness level; however, we follow a strategy for raising it. Our strategy involves hiding complex functionalities behind layers and enabling them only on demand and implementing simpler and restricted forms of them. For example, we consider implementing negation and disjunction on the data property level rather than the object property level due to the visual simplicity of the former.

Domain experts also enquired about the ability to combine and join concepts that are not directly related, and automatic filtering of attributes and attribute values depending on the constraints selected earlier and the query formulated so far. Regarding the former, one could consider two forms of such functionality. First, a user could drop several concepts into the query pane and expect the system to connect them, which is called *multi-pivot* approach [71]. Second, a user could join the pivot with a distant concept that is multiple hops away [11]; we call this *non-local navigation* [91]. In

both cases, the problem is suggesting appropriate connecting relationships. An often used approach is finding the shortest paths between the given set of concepts [11,71]; however, a shortest path approach does not necessarily lead to semantically meaningful connections. A better approach would be extending our ranking approach to offer paths. Considering automatic filtering of property values, this is among our current efforts; it is analogous to faceted search and requires frequent interaction with the data to calculate partial results, which could be problematic for large data sources.

Scalability against large ontologies in terms of user interaction [46] (i.e., large amount of properties and concepts to choose from) also requires further work. Currently, we tackle visual scalability through gradual and on-demand access to terminological knowledge and ranking concepts and properties with respect to the partial query at hand and the query log. However, we did not get a chance to evaluate its effectiveness as it requires collecting data over longer terms before experimenting with the users, while query logs from public endpoints are either not of high quality or are not complex enough structurally to detect patterns. Complementary approaches could include using modularisation techniques [23,75] with respect to interests of different user groups and letting users hide concepts and ontology elements they are not interested in their daily routine.

Finally, the quality of the ontology and supplementary information (e.g., descriptions, naming, icons etc.) is also important. Users often get confused when an ontology does not represent their understanding of the domain, and the namings and descriptions etc. are unclear or misleading. An ontology should not be considered just as an input to a work environment, it is also an output representing a collective and explicit understanding of the domain, that is, it is essentially owned by its users. However, in organisations, creating ontologies from scratch is a painful and long process especially when there is a lack of expertise; therefore, it is often preferred to automatically generate ontologies from existing schemas, which often yields poor results due to problems origination from *impedance mismatch* between relational databases and ontologies. An important contribution would be letting users provide feedback and supplementary information from the query interface.

11. Conclusion and Future Work

In this article, we have proposed an interactive end-user visual query formulation tool OptiqueVQS that is based on pragmatically motivated requirements and relies on a theoretical framework of navigation graphs. OptiqueVQS is targeted for addressing complex and specific information needs without demanding any specialised IT background and aims at providing a good balance between usability and expressivity. We evaluated our solution with different user groups, including two industrial use cases, with limited IT knowledge and skills with encouraging results.

Future work involves exploring possibilities for realising more complex query types, hidden at different layers, and supportive features such as non-local navigation and faceted search like filtering. We also plant to execute further user studies to validate the core design choices behind OptiqueVQS. Finally, we will continue working on a ranking approach and potential means to test it both from computational and user perspectives (i.e., perceived usefulness).

References

[1] Lisa Abele, Christoph Legat, Stephan Grimm, and Andreas W. Muller. Ontology-based validation of plant models. In *11th IEEE International Conference on Industrial Informatics, INDIN 2013, Bochum, Germany, July 29-31, 2013*, pages 236–241. IEEE, 2013. DOI https://doi.org/10.1109/INDIN.2013.6622888.

[2] Lisa Abele, Stephan Grimm, Sonja Zillner, and Martin Kleinsteuber. An ontology-based approach for decentralized monitoring and diagnostics. In *12th IEEE International Conference on Industrial Informatics, INDIN 2014, Porto Alegre, RS, Brazil, July 27-30, 2014*, pages 706–712. IEEE, 2014. DOI https://doi.org/10.1109/INDIN.2014.6945600.

[3] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL http://webdam.inria.fr/Alice/.

[4] Oszkár Ambrus, Knud Möller, and Siegfried Handschuh. Konduit VQB: A visual query builder for SPARQL on the social semantic desktop. In Siegfried Handschuh, Tom Heath, VinhTuan Thai, Ian Dickinson, Lora Aroyo, and Valentina Presutti, editors, *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010), Hong Kong, China, February 7, 2010*, volume 565 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010. URL http://ceur-ws.org/Vol-565/paper4.pdf.

[5] José Arancón, Luis Polo, Diego Berrueta, François-Marie Lesaffre, Nicolàs Abajo, and Antonio M. Campos. Ontology-based knowledge management in the steel industry. In Jorge S. Cardoso, Martin Hepp, and Miltiadis D. Lytras, editors, *The Semantic Web: Real-World Applications from Industry*, volume 6 of *Semantic Web and Beyond: Computing*

for Human Experience, pages 243–272. Springer, 2007. DOI https://doi.org/10.1007/978-0-387-48531-7_11.

[6] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarunas Marciuska, and Dmitriy Zheleznyakov. Faceted search over ontology-enhanced RDF data. In Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang, editors, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 939–948. ACM, 2014. DOI https://doi.org/10.1145/2661829.2662027.

[7] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarunas Marciuska, and Dmitriy Zheleznyakov. Faceted search over RDF-based knowledge graphs. *Journal of Web Semantics*, 37-38:55–74, 2016. DOI https://doi.org/10.1016/j.websem.2015.12.002.

[8] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[9] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. URL http://www.dcc.ufmg.br/irbook/.

[10] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-SPARQL: SPARQL for continuous querying. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 1061–1062. ACM, 2009. DOI https://doi.org/10.1145/1526709.1526856.

[11] Guntis Barzdins, Edgars Liepins, Marta Veilande, and Martins Zviedris. Ontology enabled graphical database query tool for end-users. In Hele Mai Haav and Ahto Kalja, editors, *Databases and Information Systems V: Selected Papers from the Eighth International Baltic Conference, DB&IS 2008*, volume 187 of *Frontiers in Artificial Intelligence and Applications*, pages 105–116. IOS Press, 2009. DOI https://doi.org/10.3233/978-1-58603-939-4-105.

[12] Konstantina Bereta, Panayiotis Smeros, and Manolis Koubarakis. Representation and querying of valid time of triples in linked geospatial data. In Philipp Cimiano, Óscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, volume 7882 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2013. DOI https://doi.org/10.1007/978-3-642-38288-8_18.

[13] Nigel Bevan and Miles MacLeod. Usability measurement in context. *Behaviour & Information Technology*, 13(1-2):132–145, 1994. DOI https://doi.org/10.1080/01449299408914592.

[14] Carlos Bobed, Guillermo Esteban, and Eduardo Mena. Enabling keyword search on Linked Data repositories: An ontology-based approach. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 17 (1):67–77, 2013. DOI https://doi.org/10.3233/KES-130255.

[15] Josep Maria Brunetti, Roberto García González, and Sören Auer. From overview to facets and pivoting for interactive exploration of Semantic Web data. *International Journal on Semantic Web and Information Systems*, 9(1):1–20, 2013. DOI https://doi.org/10.4018/jswis.2013010101.

[16] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors. *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer, 2007. DOI https://doi.org/10.1007/978-3-540-72079-9.

[17] Jean-Paul Calbimonte, Óscar Corcho, and Alasdair J. G. Gray. Enabling ontology-based access to streaming data sources. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2010. DOI https://doi.org/10.1007/978-3-642-17746-0_7.

[18] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8 (3):471–487, 2017. DOI https://doi.org/10.3233/SW-160217.

[19] Stéphane Campinas, Thomas Perry, Diego Ceccarelli, Renaud Delbru, and Giovanni Tummarello. Introducing RDF graph summary with application to assisted SPARQL formulation. In Abdelkader Hameurlain, A Min Tjoa, and Roland Wagner, editors, *23rd International Workshop on Database and Expert Systems Applications, DEXA 2012, Vienna, Austria, September 3-7, 2012*, pages 261–266. IEEE Computer Society, 2012. DOI https://doi.org/10.1109/DEXA.2012.38.

[20] Tiziana Catarci. What happened when database researchers met usability. *Information Systems*, 25(3):177–212, 2000. DOI https://doi.org/10.1016/S0306-4379(00)00015-6.

[21] Tiziana Catarci, Maria Francesca Costabile, Stefano Levialdi, and Carlo Batini. Visual query systems for databases: A survey. *Journal of Visual Languages and Computing*, 8(2):215–260, 1997. DOI https://doi.org/10.1006/jvlc.1997.0037.

[22] Cristina Civili, Marco Console, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Lorenzo Lepore, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, Valerio Santarelli, and Domenico Fabio Savo. MASTRO STUDIO: managing ontology-based data access applications. *Proceedings of the VLDB Endowment*, 6 (12):1314–1317, 2013. URL http://www.vldb.org/pvldb/vol6/p1314-poggi.pdf.

[23] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Just the right amount: extracting modules from ontologies. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 717–726. ACM, 2007. DOI https://doi.org/10.1145/1242572.1242669.

[24] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter F. Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, 2008. DOI https://doi.org/10.1016/j.websem.2008.05.001.

[25] Bernardo Cuenca Grau, Martin Giese, Ian Horrocks, Thomas Hubauer, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Michael Schmidt, Ahmet Soylu, and Dmitriy Zheleznyakov. Towards query formulation, query-driven ontology extensions in OBDA systems. In Mariano Rodriguez-Muro, Simon Jupp, and Kavitha Srinivas, editors, *Proceedings of the 10th In-*

ternational Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013., volume 1080 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013. URL http://ceur-ws.org/Vol-1080/owled2013_19.pdf.

[26] Danica Damljanovic, Milan Agatonovic, Hamish Cunningham, and Kalina Bontcheva. Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. *Journal of Web Semantics*, 19:1–21, 2013. DOI https://doi.org/10.1016/j.websem.2013.02.002.

[27] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001. DOI https://doi.org/10.1007/s007790170019.

[28] Ugo Di Giammatteo, Manuela Sagona, and Sergio Perelli. The TELEIOS software architecture - Phase II. Deliverable, FP7-257662, 2012. URL http://www.earthobservatory.eu/deliverables/FP7-257662-TELEIOS-D1.2.2.pdf.

[29] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012. URL http://research.cs.wisc.edu/dibook/.

[30] Richard G. Epstein. The TableTalk query language. *Journal of Visual Languages and Computing*, 2(2):115–141, 1991. DOI https://doi.org/10.1016/S1045-926X(05)80026-6.

[31] Martin Giese, Diego Calvanese, Peter Haase, Ian Horrocks, Yannis Ioannidis, Heralk Kllapi, Manolis Koubarakis, Maurizio Lenzerini, Ralf Möller, Mariano Rodriguez-Muro, Özgur Özçep, Riccardo Rosati, Rudolf Schlatte, Michael Schmidt, Ahmet Soylu, and Arild Waaler. Scalable end-user access to big data. In Rajendra Akerkar, editor, *Big Data Computing*, chapter 6, pages 205–244. CRC Press, 2013.

[32] Martin Giese, Ahmet Soylu, Guillermo Vega-Gorgojo, Arild Waaler, Peter Haase, Ernesto Jiménez-Ruiz, Davide Lanti, Martín Rezk, Guohui Xiao, Özgür L. Özçep, and Riccardo Rosati. Optique: Zooming in on big data. *IEEE Computer*, 48(3):60–67, 2015. DOI https://doi.org/10.1109/MC.2015.82.

[33] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014. DOI https://doi.org/10.1007/s10817-014-9305-1.

[34] Alfio Gliozzo, Or Biran, Siddharth Patwardhan, and Kathleen McKeown. Semantic technologies in IBM Watson. In Ivan Derzhanski and Dragomir Radev, editors, *Proceedings of the Fourth Workshop on Teaching Natural Language Processing, August 9, 2013 Sofia, Bulgaria*, pages 85–92, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3413.

[35] Irlán Grangel-González, Lavdim Halilaj, Gökhan Coskun, Sören Auer, Diego Collarana, and Michael Hoffmeister. Towards a semantic administrative shell for Industry 4.0 components. In *Tenth IEEE International Conference on Semantic Computing, ICSC 2016, Laguna Hills, CA, USA, February 4-6, 2016*, pages 230–237. IEEE Computer Society, 2016. DOI https://doi.org/10.1109/ICSC.2016.58.

[36] Jinghua Groppe, Sven Groppe, and Andreas Schleifer. Visual query system for analyzing social semantic web. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011 (Companion Volume)*, pages 217–220. ACM, 2011. DOI https://doi.org/10.1145/1963192.1963293.

[37] Florian Haag, Steffen Lohmann, Steffen Bold, and Thomas Ertl. Visual SPARQL querying based on extended filter/flow graphs. In Paolo Paolini and Franca Garzotto, editors, *International Working Conference on Advanced Visual Interfaces, AVI 2014, Como, Italy, May 27-29, 2014*, pages 305–312. ACM, 2014. DOI https://doi.org/10.1145/2598153.2598185.

[38] Florian Haag, Steffen Lohmann, Stephan Siek, and Thomas Ertl. QueryVOWL: A visual query notation for linked data. In Fabien Gandon, Christophe Guéret, Serena Villata, John G. Breslin, Catherine Faron-Zucker, and Antoine Zimmermann, editors, *The Semantic Web: ESWC 2015 Satellite Events - ESWC 2015 Satellite Events Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, volume 9341 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2015. DOI https://doi.org/10.1007/978-3-319-25639-9_51.

[39] Peter Haase, Michael Schmidt, and Andreas Schwarte. The information workbench as a self-service platform for linked data applications. In Olaf Hartig, Andreas Harth, and Juan F. Sequeda, editors, *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*, volume 782 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011. URL http://ceur-ws.org/Vol-782/HaaseEtAl_COLD2011.pdf.

[40] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. W3C Recommendation, 21 March 2013. URL http://www.w3.org/TR/sparql11-query/.

[41] Philipp Heim and Jürgen Ziegler. Faceted visual exploration of semantic data. In Achim Ebert, Alan J. Dix, Nahum D. Gershon, and Margit Pohl, editors, *Human Aspects of Visualization - Second IFIP WG 13.7 Workshop on Human-Computer Interaction and Visualization, HCIV (INTERACT) 2009, Uppsala, Sweden, August 24, 2009, Revised Selected Papers*, volume 6431 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2009. DOI https://doi.org/10.1007/978-3-642-19641-6_5.

[42] Aidan Hogan, Andreas Harth, Jürgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker. Searching and browsing linked data with SWSE: the semantic web search engine. *Journal of Web Semantics*, 9(4):365–401, 2011. DOI https://doi.org/10.1016/j.websem.2011.06.004.

[43] Frederik Hogenboom, Viorel Milea, Flavius Fransincar, and Uzay Kaymak. RDF-GL: A SPARQL-based graphical query language for RDF. In Richard Chbeir, Youakim Badr, Ajith Abraham, and Aboul Ella Hassanien, editors, *Emergent Web Intelligence: Advanced Information Retrieval*, Advanced Information and Knowledge Processing, pages 87–116. Springer, 2010. DOI https://doi.org/10.1007/978-1-84996-074-8_4.

[44] Matthew Horridge, Nick Drummond, John Goodwin, Alan L. Rector, Robert Stevens, and Hai Wang. The Manchester OWL syntax. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006. URL http://ceur-ws.org/Vol-216/submission_9.pdf.

[45] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible SROIQ. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Repre-*

sentation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006, pages 57–67. AAAI Press, 2006. URL http://www.aaai.org/Library/KR/2006/kr06-009.php.

[46] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia G. Giannopoulou. Ontology visualization methods - a survey. *ACM Computing Surveys*, 39(4):10, 2007. DOI https://doi.org/10.1145/1287620.1287621.

[47] Esther Kaufmann and Abraham Bernstein. Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Journal Web Semantics*, 8(4):377–393, 2010. DOI https://doi.org/10.1016/j.websem.2010.06.001.

[48] Ali Khalili and Sören Auer. User interfaces for semantic authoring of textual content: A systematic literature review. *Journal of Web Semantics*, 22:1–18, 2013. DOI https://doi.org/10.1016/j.websem.2013.08.004.

[49] Evgeny Kharlamov, Ernesto Jiménez-Ruiz, Dmitriy Zheleznyakov, Dimitris Bilidas, Martin Giese, Peter Haase, Ian Horrocks, Herald Kllapi, Manolis Koubarakis, Özgür L. Özçep, Mariano Rodriguez-Muro, Riccardo Rosati, Michael Schmidt, Rudolf Schlatte, Ahmet Soylu, and Arild Waaler. Optique: Towards OBDA systems for industry. In Philipp Cimiano, Miriam Fernández, Vanessa López, Stefan Schlobach, and Johanna Völker, editors, *The Semantic Web: ESWC 2013 Satellite Events - ESWC 2013 Satellite Events, Montpellier, France, May 26-30, 2013, Revised Selected Papers*, volume 7955 of *Lecture Notes in Computer Science*, pages 125–140. Springer, 2013. DOI https://doi.org/10.1007/978-3-642-41242-4_11.

[50] Evgeny Kharlamov, Nina Solomakhina, Özgür Lütfü Özçep, Dmitriy Zheleznyakov, Thomas Hubauer, Steffen Lamparter, Mikhail Roshchin, Ahmet Soylu, and Stuart Watson. How semantic technologies can enhance data access at Siemens Energy. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 601–619. Springer, 2014. DOI https://doi.org/10.1007/978-3-319-11964-9_38.

[51] Evgeny Kharlamov, Dag Hovland, Ernesto Jiménez-Ruiz, Davide Lanti, Hallstein Lie, Christoph Pinkel, Martín Rezk, Martin G. Skjæveland, Evgenij Thorstensen, Guohui Xiao, Dmitriy Zheleznyakov, and Ian Horrocks. Ontology based access to exploration data at Statoil. In Marcelo Arenas, Óscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul T. Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 93–112. Springer, 2015. DOI https://doi.org/10.1007/978-3-319-25010-6_6.

[52] Evgeny Kharlamov, Ernesto Jiménez-Ruiz, Christoph Pinkel, Martin Rezk, Martin G. Skjæveland, Ahmet Soylu, Guohui Xiao, Dmitriy Zheleznyakov, Martin Giese, Ian Horrocks, and Arild Waaler. Optique: Ontology-based data access platform. In Serena Villata, Jeff Z. Pan, and Mauro Dragoni, editors, *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.*, volume 1486 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. URL http://ceur-ws.org/Vol-1486/paper_24.pdf.

[53] Evgeny Kharlamov, Sebastian Brandt, Ernesto Jiménez-Ruiz, Yannis Kotidis, Steffen Lamparter, Theofilos Mailis, Christian Neuenstadt, Özgür L. Özçep, Christoph Pinkel, Christoforos Svingos, Dmitriy Zheleznyakov, Ian Horrocks, Yannis E. Ioannidis, and Ralf Möller. Ontology-based integration of streaming and static relational data with Optique. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 2109–2112. ACM, 2016. DOI https://doi.org/10.1145/2882903.2899385.

[54] Evgeny Kharlamov, Bernardo Cuenca Grau, Ernesto Jiménez-Ruiz, Steffen Lamparter, Gulnar Mehdi, Martin Ringsquandl, Yavor Nenov, Stephan Grimm, Mikhail Roshchin, and Ian Horrocks. Capturing industrial information models with ontologies and constraints. In Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, volume 9982 of *Lecture Notes in Computer Science*, pages 325–343, 2016. DOI https://doi.org/10.1007/978-3-319-46547-0_30.

[55] Evgeny Kharlamov, Yannis Kotidis, Theofilos Mailis, Christian Neuenstadt, Charalampos Nikolaou, Özgür L. Özçep, Christoforos Svingos, Dmitriy Zheleznyakov, Sebastian Brandt, Ian Horrocks, Yannis E. Ioannidis, Steffen Lamparter, and Ralf Möller. Towards analytics aware ontology based access to static and streaming data. In Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, volume 9982 of *Lecture Notes in Computer Science*, pages 344–362, 2016. DOI https://doi.org/10.1007/978-3-319-46547-0_31.

[56] Evgeny Kharlamov, Dimitris Bilidas, Dag Hovland, Ernesto Jimenez-Ruiz, Davide Lanti, Hallstein Lie, Martin Rezk, Martin G. Skjæveland, Ahmet Soylu, Guohui Xiao, Dmitriy Zheleznyakov, Martin Giese, Yannis Ioannidis, Yannis Kotidis, Manolis Koubarakis, and Arild Waaler. Ontology based data access in Statoil. *Journal of Web Semantics*, 44:3–36, 2017. DOI https://doi.org/10.1016/j.websem.2017.05.005.

[57] Evgeny Kharlamov, Theofilos Mailis, Gulnar Mehdi, Christian Neuenstadt, Ozgur Ozcep, Mikhail Roshchin, Nina Solomakhina, Ahmet Soylu, Christoforos Svingos, Sebastian Brandt, Martin Giese, Yannis Ioannidis, Steffen Lamparter, Ralf Moller, Yannis Kotidis, and Arild Waaler. Semantic access to streaming and static data at Siemens. *Journal of Web Semantics*, 44:54–74, 2017. DOI https://doi.org/10.1016/j.websem.2017.02.001.

[58] Mikhail R. Kogalovsky. Ontology-based data access systems. *Programming and Computer Software*, 38(4):167–182, 2012. DOI https://doi.org/10.1134/S0361768812040032.

[59] Y. Tina Lee. Information modeling: From design to implementation. In *Proceedings of the Second World Manufactur-*

ing Congress (WMC 1999), September 27-30, 1999, Durham, UK., pages 315–321, 1999. URL http://www.mel.nist.gov/msidlibrary/doc/confout.pdf.

[60] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-user development: An emerging paradigm. In Henry Lieberman, Fabio Paternò, and Volker Wulf, editors, *End User Development*, Human-Computer Interaction Series, pages 1–8. Springer, 2006. DOI https://doi.org/10.1007/1-4020-5386-X_1.

[61] Vanessa López, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Journal of Web Semantics*, 21:3–13, 2013. DOI https://doi.org/10.1016/j.websem.2013.05.006.

[62] Jaime I. Lopez-Veyna, Víctor Jesús Sosa Sosa, and Ivan López-Arévalo. KESOSD: keyword search over structured data. In Tok Wang Ling, Ge Yu, Jiaheng Lu, and Wei Wang, editors, *Proceedings of the Third International Workshop on Keyword Search on Structured Data, KEYS 2012, Scottsdale, AZ, USA, May 20, 2012*, pages 23–31. ACM, 2012. DOI https://doi.org/10.1145/2254736.2254743.

[63] Gary Marchionini and Ryen White. Find what you need, understand what you find. *International Journal of Human-Computer Interaction*, 23(3):205–237, 2007. DOI https://doi.org/10.1080/10447310701702352.

[64] Boris Motik, Peter Patel-Schneider, and Bernardo Cuenca Grau. *OWL 2 Web Ontology Language: Direct Semantics (Second Edition)*. W3C Recommendation, 11 December 2012. URL http://www.w3.org/TR/owl2-direct-semantics/.

[65] Charalampos Nikolaou, Kallirroi Dogani, Konstantina Bereta, George Garbis, Manos Karpathiotakis, Kostis Kyzirakos, and Manolis Koubarakis. Sextant: Visualizing time-evolving linked geospatial data. *Journal of Web Semantics*, 35:35–52, 2015. DOI https://doi.org/10.1016/j.websem.2015.09.004.

[66] Özgür Lutfu Özcep, Ralf Möller, and Christian Neuenstadt. A Stream-Temporal Query Language for Ontology Based Data Access. In *Proceedings of the 37th Annual German Conference on Artificial Intelligence (KI 2014)*, volume 8736 of *LNCS*, pages 183–194. Springer, 2014.

[67] Matthew Perry and John Herring, editors. *GeoSPARQL - A Geographic Query Language for RDF Data*. OGC Implementation Standard, 2012. URL http://www.opengeospatial.org/standards/geosparql.

[68] Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 370–388. Springer, 2011. DOI https://doi.org/10.1007/978-3-642-25073-6_24.

[69] François Picalausa and Stijn Vansummeren. What are real SPARQL queries like? In Roberto De Virgilio, Fausto Giunchiglia, and Letizia Tanca, editors, *Proceedings of the International Workshop on Semantic Web Information Management, SWIM 2011, Athens, Greece, June 12, 2011*, page 7. ACM, 2011. DOI https://doi.org/10.1145/1999299.1999306.

[70] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *Journal on Data Semantics*, 10:133–173, 2008. DOI https://doi.org/10.1007/978-3-540-77688-8_5.

[71] Igor O. Popov, Monica M. C. Schraefel, Wendy Hall, and Nigel Shadbolt. Connecting the dots: A multi-pivot approach to data exploration. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 553–568. Springer, 2011. DOI https://doi.org/10.1007/978-3-642-25073-6_35.

[72] Robin G. Qiu and Mengchu Zhou. Mighty MESs; state-of-the-art and future manufacturing execution systems. *IEEE Robotics & Automation Magazine*, 11(1):19–25, 2004. DOI https://doi.org/10.1109/MRA.2004.1275947.

[73] Hoan Nguyen Mau Quoc, Martin Serrano, Danh Le Phuoc, and Manfred Hauswirth. Super Stream Collider: Linked stream mashups for everyone. In *Proceedings of the Semantic Web Challenge 2012 at 11th International Semantic Web Conference (ISWC 2012)*, 2012.

[74] Martin Ringsquandl, Steffen Lamparter, Sebastian Brandt, Thomas Hubauer, and Raffaello Lepratti. Semantic-guided feature selection for industrial automation systems. In Marcelo Arenas, Óscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul T. Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 225–240. Springer, 2015. DOI https://doi.org/10.1007/978-3-319-25010-6_13.

[75] Ana Armas Romero, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. Module extraction in expressive ontology languages via datalog reasoning. *Journal of Artificial Intelligence Research*, 55:499–564, 2016. DOI https://doi.org/10.1613/jair.4898.

[76] Monica M. C. Schraefel, Max Wilson, Alistair Russell, and Daniel A. Smith. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM*, 49(4):47–49, 2006. DOI https://doi.org/10.1145/1121949.1121980.

[77] Ben Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983. DOI https://doi.org/10.1109/MC.1983.1654471.

[78] Martin G. Skjæveland, Martin Giese, Dag Hovland, Espen H. Lian, and Arild Waaler. Engineering ontology-based access to real-world data sources. *Journal of Web Semantics*, 33:112–140, 2015. DOI https://doi.org/10.1016/j.websem.2015.03.002.

[79] Paul R. Smart, Alistair Russell, Dave Braines, Yannis Kalfoglou, Jie Bao, and Nigel R. Shadbolt. A visual approach to semantic query design using a web-based graphical query designer. In Aldo Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, volume 5268 of *Lecture Notes in Computer Science*, pages 275–291. Springer, 2008. DOI https://doi.org/10.1007/978-3-540-87696-0_25.

[80] Ahmet Soylu and Martin Giese. Qualifying ontology-based visual query formulation. In Troels Andreasen, Henning Christiansen, Janusz Kacprzyk, Henrik Legind Larsen, Gabriella Pasi, Olivier Pivert, Guy De Tré, M. Amparo Vila, Adnan Yazici, and Slawomir Zadrozny, editors, *Flexible Query Answering Systems 2015 - Proceedings of the 11th International Conference FQAS 2015, Cracow, Poland, October 26-28, 2015*, volume 400 of *Advances in Intelligent Systems and Computing*, pages 243–255. Springer, 2015. DOI https://doi.org/10.1007/978-3-319-26154-6_19.

[81] Ahmet Soylu, Patrick De Causmaecker, and Piet Desmet. Context and adaptivity in pervasive computing environments: Links with software engineering and ontological engineering. *Journal of Software*, 4(9):992–1013, 2009. DOI https://doi.org/10.4304/jsw.4.9.992-1013.

[82] Ahmet Soylu, Patrick De Causmaecker, Davy Preuveneers, Yolande Berbers, and Piet Desmet. Formal modelling, knowledge representation and reasoning for design and development of user-centric pervasive software: A meta-review. *International Journal of Metadata, Semantics and Ontologies*, 6(2):96–125, 2011. DOI https://doi.org/10.1504/IJMSO.2011.046595.

[83] Ahmet Soylu, Felix Mödritscher, and Patrick De Causmaecker. Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. *Integrated Computer-Aided Engineering*, 19(1):93–109, 2012. DOI https://doi.org/10.3233/ICA-2012-0393.

[84] Ahmet Soylu, Felix Mödritscher, Fridolin Wild, Patrick De Causmaecker, and Piet Desmet. 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. DOI https://doi.org/10.1108/00330331211276486.

[85] Ahmet Soylu, Martin Giese, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, and Ian Horrocks. OptiqueVQS: Towards an ontology-based visual query system for big data. In Latif Ladid, Antonio Montes, Peter A. Bruck, Fernando Ferri, and Richard Chbeir, editors, *Fifth International Conference on Management of Emergent Digital EcoSystems, MEDES '13, Luxembourg, Luxembourg, October 29-31, 2013*, pages 119–126. ACM, 2013. DOI https://doi.org/10.1145/2536146.2536149.

[86] Ahmet Soylu, Martin G. Skjæveland, Martin Giese, Ian Horrocks, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, and Dmitriy Zheleznyakov. A preliminary approach on ontology-based visual query formulation for big data. In Emmanouel Garoufallou and Jane Greenberg, editors, *Metadata and Semantics Research - 7th Research Conference, MTSR 2013, Thessaloniki, Greece, November 19-22, 2013. Proceedings*, volume 390 of *Communications in Computer and Information Science*, pages 201–212. Springer, 2013. DOI https://doi.org/10.1007/978-3-319-03437-9_21.

[87] Ahmet Soylu, Martin Giese, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, and Ian Horrocks. Towards exploiting query history for adaptive ontology-based visual query formulation. In Sissi Closs, Rudi Studer, Emmanouel Garoufallou, and Miguel-Ángel Sicilia, editors, *Metadata and Semantics Research - 8th Research Conference, MTSR 2014, Karlsruhe, Germany, November 27-29, 2014. Proceedings*, volume 478 of *Communications in Computer and Information Science*, pages 107–119. Springer, 2014. DOI https://doi.org/10.1007/978-3-319-13674-5_11.

[88] Ahmet Soylu, Evgeny Kharlamov, Dmitriy Zheleznyakov, Ernesto Jiménez-Ruiz, Martin Giese, and Ian Horrocks. Ontology-based visual query formulation: An industry experience. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ioannis T. Pavlidis, Rogério Schmidt Feris, Tim McGraw, Mark Elendt, Regis Kopper, Eric D. Ragan, Zhao Ye, and Gunther H. Weber, editors, *Advances in Visual Computing - 11th International Symposium, ISVC 2015, Las Vegas, NV, USA, December 14-16, 2015, Proceedings, Part I*, volume 9474 of *Lecture Notes in Computer Science*, pages 842–854. Springer, 2015. DOI https://doi.org/10.1007/978-3-319-27857-5_75.

[89] Ahmet Soylu, Martin Giese, Ernesto Jiménez-Ruiz, Guillermo Vega-Gorgojo, and Ian Horrocks. Experiencing OptiqueVQS: A multi-paradigm and ontology-based visual query system for end users. *Universal Access in the Information Society*, 15(1):129–152, 2016. DOI https://doi.org/10.1007/s10209-015-0404-5.

[90] Ahmet Soylu, Martin Giese, Rudolf Schlatte, Ernesto Jiménez-Ruiz, Özgür L. Özçep, and Sebastian Brandt. Domain experts surfing on stream sensor data over ontologies. In Thanos G. Stavropoulos, Georgios Meditskos, and Antonis Bikakis, editors, *Proceedings of the 1st Workshop on Semantic Web Technologies for Mobile and Pervasive Environments co-located with the 13th Extended Semantic Web Conference (ESWC 2016), Heraklion, Greece, May 29, 2016.*, volume 1588 of *CEUR Workshop Proceedings*, pages 11–20. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1588/paper4.pdf.

[91] Ahmet Soylu, Martin Giese, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, and Ian Horrocks. Ontology-based end-user visual query formulation: Why, what, who, how, and which? *Universal Access in the Information Society*, 16(2):435–467, 2017. DOI https://doi.org/10.1007/s10209-016-0465-0.

[92] Ahmet Soylu, Martin Giese, Rudolf Schlatte, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Özgür L. Özçep, Christian Neuenstadt, and Sebastian Brandt. Querying industrial stream-temporal data: An ontology-based visual approach. *Journal of Ambient Intelligence and Smart Environments*, 9 (1):77–95, 2017. DOI https://doi.org/10.3233/AIS-160415.

[93] Kent Spackman. SNOMED RT and SNOMED CT. Promise of an international clinical ontology. *M.D. Computing*, 17 (6):29, 2000. URL https://www.ncbi.nlm.nih.gov/labs/articles/11189756/.

[94] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web*, 3(2):169–209, 2012. DOI https://doi.org/10.3233/SW-2011-0055.

[95] Bongwon Suh and Benjamin B. Bederson. OZONE: a zoomable interface for navigating ontology information. In Maria De Marsico, Stefano Levialdi, and Emanuele Panizzi, editors, *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2002, Trento, Italy, May 22-24, 2002*, pages 139–143. ACM, 2002. DOI https://doi.org/10.1145/1556262.1556284.

[96] Alistair G. Sutcliffe. Evaluating the costs and benefits of end-user development. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–4, 2005. DOI https://doi.org/10.1145/1082983.1083241.

[97] Arthur H. M. ter Hofstede, Henderik Alex Proper, and Theo P. van der Weide. Query formulation as an information retrieval problem. *The Computer Journal*, 39(4):255–274, 1996. DOI https://doi.org/10.1093/comjnl/39.4.255.

[98] Thanh Tran, Daniel M. Herzig, and Günter Ladwig. Semsearchpro - using semantics throughout the search process. *Journal of Web Semantics*, 9(4):349–364, 2011. DOI https://doi.org/10.1016/j.websem.2011.08.004.

[99] Daniel Tunkelang. *Faceted Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009. DOI https://doi.org/10.2200/S00190ED1V01Y200904ICR005.

[100] Victoria S. Uren, Yuangui Lei, Vanessa López, Haiming Liu, Enrico Motta, and Marina Giordanino. The uasability of semantic search tools: A review. *The Knowledge Engineering Review*, 22(4):361–377, 2007. DOI https://doi.org/10.1017/S0269888907001233.

[101] Cornelis Joost van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[102] Guillermo Vega-Gorgojo, Martin Giese, Simen Heggestøyl, Ahmet Soylu, and Arild Waaler. PepeSearch: semantic data for the masses. *PLoS ONE*, 11(3):1–12, 2016. DOI https://doi.org/10.1371/journal.pone.0151573.

[103] Guillermo Vega-Gorgojo, Laura Slaughter, Martin Giese, Simen Heggestøyl, Ahmet Soylu, and Arild Waaler. Visual query interfaces for semantic datasets: An evaluation study. *Journal of Web Semantics*, 39:81–96, 2016. DOI https://doi.org/10.1016/j.websem.2016.01.002.