Noname manuscript No.
(will be inserted by the editor)

Ontology-based End-user Visual Query Formulation: Why, what, who, how, and which?

Ahmet Soylu · Martin Giese · Ernesto Jimenez-Ruiz · Evgeny Kharlamov · Dmitriy Zheleznyakov · Ian Horrocks

Received: date / Accepted: date

Abstract Value creation in an organisation is a time-sensitive and data-intensive process, yet it is often delayed and bounded by the reliance on IT experts extracting data for domain experts. Hence, there is a need for providing people who are not professional developers with the flexibility to pose relatively complex and ad hoc queries in an easy and intuitive way. In this respect, visual methods for query formulation undertake the challenge of making querying independent of users' technical skills and the knowledge of the underlying textual query language and the structure of data. An ontology is more promising than the logical schema of the underlying data for guiding users in formulating queries, since it provides a richer vocabulary closer to the users' understanding. However, on the one hand, today the most of world's enterprise data reside in relational databases rather than triple stores, and on the other, visual query formulation has become more compelling due to ever-increasing data size and complexity – known as Big Data. This article presents and argues for ontology-based visual query formulation for end users; discusses its feasibility in terms of ontology-based data access, which virtualises legacy relational databases as RDF, and the dimensions of Big Data; presents key conceptual aspects and dimensions, challenges, and requirements; and reviews, categorises, and discusses notable approaches and systems.

Keywords Visual query formulation · usability · data retrieval · ontology-based data access · big data

This research is funded by the Seventh Framework Program (FP7) of the European Commission under Grant Agreement 318338, "Optique".

Ahmet Soylu
Faculty of Computer Science and Media Technology,
Norwegian University of Science and Technology
E-mail: ahmets.soylu@ntnu.no

Martin Giese
Department of Informatics, University of Oslo
E-mail: martingi@ifi.uio.no

Ernesto Jimenez-Ruiz · Evgeny Kharlamov · Dmitriy Zheleznyakov · Ian Horrocks
Department of Computer Science, University of Oxford
E-mail: {ernesto.jimenez-ruiz, evgeny.kharlamov, dmitriy.zheleznyakov, ian.horrocks}@cs.ox.ac.uk

1 Introduction

In contrast to Web search engines, *data access* in traditional database systems heavily relies on complex structures and semantics with no tolerance to irrelevant and missing results. Consequently, in order to reach precise and complete answers, *information needs* have to be specified exactly. Although approaches such as keyword-based, form-based, and faceted search work well on the Web, they per se fall short in formulating complex information needs (cf. [89]). Structured query languages such as *SQL* and *XQuery* are quite expressive, yet they require an array of technical *skills* and *knowledge* on query language, syntax and domain schema. More precisely, they require users to recall relevant schema and syntax elements and to communicate their information needs in a programmatic way. Such an approach makes database systems almost, if not completely, inaccessible to the *end users*, who do not possess necessary technical skills and knowledge (cf. [112]). For instance, in an enterprise setting, this results in a situation where a group of *IT experts* acts as an intermediary in *query formulation* (aka *query construction*) process between *domain experts* and databases. This comes with a cost of turnaround time measured in hours to days aside from the direct cost of IT experts. Thus, the challenge is to enable end users to construct queries on their own, by using the higher level of abstractions, instead of trying to articulate and explain their information needs to IT experts.

In this respect, *direct manipulation* [150], as a human-computer interaction style, has a profound impact on the successor query systems and languages. Direct manipulation employs *recognition*, rather than *recall*, and direct manipulation objects, rather than a command language syntax, to lead to easy to use and intuitive interactive systems. In this pursuit, in database domain, *visual query formulation* approaches (cf. [37,61]) exploit the high bandwidth of visual interfaces through the visual representations of domain of interest and information needs. The matter is largely one of *usability* (cf. [89, 36]), albeit the underlying formal semantics have a role to play. A good deal of research that adopts various visual representations as well as interaction styles has been carried out (cf. [37]). The literature reports varying degrees of success including different factors such as user type and the level of expressiveness. However, existing approaches suffer from the gap between the low-level domain models, such as database schemas, and end users' understanding of reality. In this respect, ontologies stand out as a natural communication medium between end users and databases (cf. [154,145]).

Nevertheless, there are still issues standing against the proliferation of ontology-based visual query formulation. First, today, most of the world's enterprise data reside in relational databases, and migrating data to triple stores is not an option mainly due to legacy applications built on top of existing relational databases. *Ontology-based data access* (OBDA) to relational data sources (cf. [101,164]) has emerged as a promising response, as data stays where it is by *virtualising* relational databases as RDF. Secondly, with the emergence of the *Big Data* phenomenon (cf. [107,117]), the challenge has already been exacerbated dramatically in terms of *scalability.* The scalability is considered in terms of human-computer dialog (i.e., *query formulation*) and performance (i.e., *query evaluation*). This is because Big Data does not only concern the performance, but also the usability of database systems, as larger and complex schemas with the current visual query formulation approaches force the limits of human visual channel and cognitive capacity.

This article aims to offer a comprehensive look into ontology-based visual query formulation and reveal insights for the development of successor systems. Specifically, the article answers and elaborates on *a*) *why* visual query formulation and ontologies (Sect. 2); *b*) *what* visual query formulation is and its core aspects and dimensions (Sect. 3); *c*) *who* benefits most from such systems (Sect. 3); *d*) *how* feasible it is given a landscape dominated by relational databases (Sect. 4); *e*) *which* specific challenges and requirements there are (Sect. 5); *f*) *which* approaches exist along with their pros and cons (Sect. 6); and, *g*) *which* research directions should be considered by practitioners and researchers (Sect. 6).

2 Motivation

The primary aim of an organisation is creating value such as financial, social, cognitive, and political. The role of information technology is indispensable particularly in the success of data intensive *value creation* processes such as *decision-making*, *sense-making*, and *intelligence analysis* (cf. [132,134,68]). However, there is often a large gap between the people who interpret and use data and IT systems where data is stored and processed. This is mostly because *a*) although domain experts are provided with data analysis tools, they often lack end-user tools for extracting data from databases, and *b*) there is often no high level conceptual vocabulary for end-users to formulate their information needs – database schemas are rather low level artefacts pertaining to implementation of the actual systems.

2.1 Data access bottleneck

Nunamaker et al. [132] break value creation efforts into a tripartite model that capitalises on information technology tools. The model consists of *information assimilation*, *group dynamics*, and *methodology.* Each part builds on and contributes to each other. Information assimilation addresses concerns regarding finding and understanding information (i.e., data and communication infrastructure), while group dynamics concern the *collaboration* of group members. The information assimilation and group dynamics together form an *intellectual bandwidth* that bounds the value creation potential of an organisation. The third part, methodology, addresses the steps and guidelines required to leverage the intellectual bandwidth for value creation (i.e., reason and action). Apparently, data access (i.e., information assimilation) is a part of a larger process [81], yet it is a principal component of intellectual bandwidth and can easily turn into a bottleneck. In data intensive industries (e.g., [85,121]), end users, who are domain experts who need to extract data from databases in their expertise domain [10], spend up to 80% of their time on data access problems (cf. [69]).

This data access bottleneck is mostly due to the sharp distinction between employees who have technical skills and knowledge to extract data (i.e., database/IT

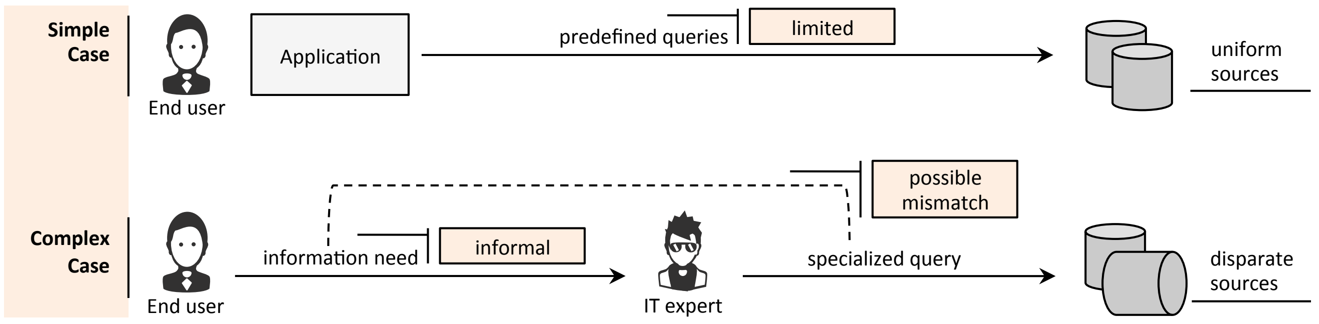


Fig. 1 Traditional data access scenarios.

experts, skilled users) and those who have domain knowledge and know how to interpret and use data (i.e., domain experts). End users cannot interact with databases directly and are unable to pose complex queries against data sources on their own, since they usually do not possess necessary skills and knowledge on the database structure and query language. End users are dependent on IT experts, who either provide a set of predefined queries built into applications and/or collaborate with the end users to develop ad-hoc queries – see Figure 1. In both cases, decision support activities are hampered by data access problems. This is because, on the one hand, the use of predefined queries is inflexible, since it is not possible to anticipate every information need in advance. On the other hand, collaboration between end users and IT experts is mostly based on the vague descriptions of information needs; hence, it is generally time-consuming and often results in repetitive *query reformulations*. In many cases, the cost of query reformulation is high, since a query formulation phase is often followed by a potentially lengthy *query evaluation* phase [89].

The lack of possibility to directly reach reliable data in a timely fashion indeed points to an *accessibility* challenge. Though the term "accessibility" is often linked to situations involving physical disabilities, there are more factors that inhibit people from full access to technology, and in the present context relates to knowledge and skill barriers (cf. [178]). Direct manipulation is paramount in visual query formulation and promising to remove this accessibility barrier due to the closeness of the solution domain to the reality, ability to handle large and complex information, low memory load (i.e., recall vs. recognition), and low comprehension demand/immediate grasping. These benefits potentially lead to ease in learning, high efficiency rate, reduced error rate, and satisfactory experiences (cf. [150,37]).

The visual query formulation approach eliminates the *man-in-the-middle* and allows end users to extract and use data on their own. This first moves organisations and individuals closer to fulfil their intellectual bandwidth, that is a considerable time and resources could be freed-up and could be redeployed so as to contribute value creation. Secondly, end users could augment their intellectual bandwidth as they can extract, combine, and interpret data in previously unforeseen ways.

2.2 Impedance mismatch and reasoning

Gruber [74] and Borst [25] define an ontology as a *formal, explicit specification of a shared conceptualisation*, which is an abstract representation of a phenomenon in the world based on identification of relevant *concepts* (aka *classes*), *attributes* (aka *data type properties*), *relationships* (aka *roles, associations and object properties*), and *constraints* (cf. [169]). In a more precise language, a consensual (i.e., shared) conceptualisation, constituted by a specific vocabulary and assumptions on the intended meaning (cf. [75]), becomes an ontology only when it is explicitly formulated with an artificial machine-readable language (i.e., formal). Ontologies are fundemental for the construction of *knowledge bases*, which do not only contain *terminological knowledge* such as concepts and properties but also their *instances* (aka *individuals*) as *assertional knowledge* (cf. [72]). The purpose is to express and capture the domain structure and knowledge, and to apply *reasoning* over it (i.e., to derive conclusions that are implicitly available in the knowledge base).

Ontologies differ in expressiveness from *lightweight* to *heavyweight*. A lightweight ontology, in the simplest case, describes a hierarchy of concepts with concept - subconcept relationships; a heavyweight ontology employs complex representation primitives, *axioms* (i.e., logical expressions that are always true), and constraints to express more sophisticated relationships (cf. [72,131]). The reasoning power of an ontology is constrained with the form of representation, and is mostly limited with tasks such as *subsumption* (i.e., determines concept - subconcept relationships) and *individual checking/realisation* (i.e., determines whether a given instance belongs to a particular concept). This applies to representation formalisms based on *artificial intelligence* (AI), where approaches based on *software engineering* (e.g., UML), *database engineering* (e.g., ER, EER) and other models are very weak both in terms of reasoning and expressiveness (cf. [72]).

Expressiveness and reasoning, for ontologies, are two sides of a coin, for which a trade-off exists. This is due to the fact that, while increasing the expressiveness, it has to be guaranteed that all conclusions are still computable (i.e., *completeness*) and finish in finite time (i.e., *decidability*) (cf. [111]). The combination of rules with ontologies is a well-known way to increase the reasoning power of ontologies (cf. [125,157]). There are two possible ways: one is to build rules on top of ontologies, where rules follow the vocabulary specified in the ontology; the other way is to build rules on top of ontologies, where ontological definitions are supplemented by rules (cf. [58, 166]). These originate from the fundamental differences that exist between ontologies and rules: ontologies, under the *knowledge representation* (KR) paradigm, focus on content to describe knowledge, while rules, under the *logic programming* (LP) paradigm, focus on form to arrive logical conclusions (i.e., reasoning). Therefore, the selection of representation formalism is strictly dependent on the level of support required for expressiveness and reasoning [157].

The very same angle that crosses knowledge representation and logic reveals the role that ontologies have to play not only for end-user data access, but also in usability at large. Herein, from the knowledge representation perspective, one first needs to consider the existential differences between ontologies and models to reach a profound understanding of the context. A *model* is defined as an abstraction representing some view of a reality, by necessarily omitting details, intended for a definite purpose [83], such as UML and EER models for software and database design respectively. Ruiz and Hilera [145] argue that, although there is a confusion between ontologies and models due to examples where they are represented with the same language, ontologies are *descriptive*, that is they describe what already exists, while models are *prescriptive*, that is they prescribe systems, which do not exist, and pertain to implementation of them (cf. [6,64,157,83]). In other words, models are forward looking *solution/implementation domain* artefacts, while ontologies are backward looking *problem domain* artefacts. Hence, ontologies are essentially meant to be richer and closer to end-user vocabulary and understanding than *database schemas* (cf. [38,154]). For instance, an ontology naturally connects domain concepts, while a database schema relies on unnatural flattening and scattering approach built on *normalization/join* notion (cf. [89]), leading to an *impedance mismatch.* Consequently, a visual query tool built on ontologies provides a closer representation of the problem domain, hence enabling users to understand and communicate better. This is not limited with the human-machine interaction, since ontologies have the potential to act as an unambiguous communication medium for human-human dialogs, i.e., *collaboration* (cf. group dynamics), and machine-machine dialogs, i.e., *interoperability.*

Finally, the reasoning capability brings in the power of expressing more with less both in the query formulation stage and during the answering stage by relating the whole set of implied information instead of what is explicitly stated. For instance, imagine a system where different types of users exist, such as students, lecturers, and administrative employees. A user query, "give me all the persons who live in Oslo", is at a sufficient specificity in an ontology-based system, as it can infer any student, lecturer, and employee is also a person. This includes both the form of reasoning attached to ontologies and various types of logic rules, such as the followings proposed by Boley et al. [24]: *deduction*, *normative*, and *reactive* rules to derive new information, to check the consistency and integrity of data, and to define automatic actions triggered by the occurrence of data of interest respectively. Moreover, ontologies carry a notable potential to provide *causal explanations* (cf. [157,19]), which can be used for various purposes such as for explaining the source of inferred data and inconsistencies in data and user queries.

3 Visual query formulation

The purpose of a visual query formulation tool is to facilitate retrieval of data that makes a value for an individual or organisation. Visual query formulation approaches exploit the *bi-directional* and the *multi-sensory* characteristics of visual representations. Collectively, they inform users about schema concepts, system affordances, and data, through a number of sensory variables such as *texture*, *colour*, *size*, and *shape*; and allow users to communicate their information needs by interacting with them (cf. [37,36]). The concern is essentially about the productivity and the quality of human-machine interaction, rather than the functionality or the technology of software (cf. [89,36]).

Shneiderman [150] elaborates on the *syntactic/semantic model* of user behaviour to reach an understanding of what makes an interactive system successful. The model distinguishes two types of knowledge: *syntactic* and *semantic knowledge.* The former relates to the expertise of the grammar/syntax of a system or language;

this type of knowledge is arbitrary and proprietary (e.g., command to capitalise a letter in a text editor). The latter is not system or language dependent; rather, it relates to the knowledge of meanings in terms of high-level concepts and functionality (e.g., the process of moving a sentence from one paragraph to another).

The syntactic/semantic model underpins the direct manipulation approach and aims to realise user experiences in which interactive systems are naturally embedded into the reality. This becomes possible by harvesting non-volatile semantic knowledge, which is acquired through general explanation, analogy and example, and common skills for manipulating physical objects, which are acquired at the *concrete operational stage* (7 -11 years) of human growth (cf. [150]).

3.1 Characteristics and categorisation

A visual query formulation tool is a *data retrieval* (DR) paradigm, which differs from *information retrieval* (IR) and *question answering* (QA) (cf. [139,8,102]). In DR, an information need has to be exact and complete, and is defined over a deterministic model with the aim of retrieving all and only those objects that exactly match the criteria. However, in IR and QA (for most of the question types), an information need is typically incomplete and loosely defined over a probabilistic model with the aim of retrieving relevant objects. In other words, DR systems have no tolerance for missing or irrelevant results and a single erroneous object implies a total failure; while IR and QA 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.

As far as visual DR methods are considered, Epstein [61] distinguishes between a *visual query language* (VQL) and a *visual query system* (VQS). A VQL is a *visual programming language* (cf. [32,31]), and is based on a well-defined formal semantics with a visual notation and syntax; there is an underlying textual language for which mostly a one-to-one correspondence exists. The latter is a *visual programming environment* (cf. [32,31]), and is mainly based on a system of interactions, rather than a visual formalism, which generates queries in target formal textual form. A VQS may or may not use a VQL [61]; if it does not, the system is likely to express only a subset of the underlying language.

The success of a visual query formulation tool varies with respect to the visual representation and the interaction paradigms employed in relation to the *frequency of interaction* (i.e., how often the tool is used), the *variance of query tasks* (i.e., whether tasks have a repetitive nature), and *query complexity* (i.e., various dimensions such as size, semantic complexity etc.) (cf. [37,36,12, 110]). However, in general, the realisation of certain tasks over a visual query system or a language can still be complex (i.e., with a compromise in usability), such as, for example, queries with cycles.

In any case, certain data access efforts have to be supported, that are *understanding the reality of interest* (i.e., exploration), which relates to the activities for understanding and finding schema concepts and relationships relevant to information need at hand; and *query construction*, which concerns the compilation of relevant concepts and constraints into formal information needs (i.e., queries) [37]. On these grounds, the choice of visual representation and interaction paradigms, along with underlying *metaphors*, *analogies* etc., is of primary importance.

Catarci et al. [37] classify VQSs with respect to *visual representation* and *interaction paradigms* in use – see Figure 2. This categorisation is not exhaustive and is only a fragment of a classification derived by Lohse et al. [114] for visual representations. The selected categories mostly concern visual query tools that are used to access alphanumeric data. Visual representation paradigms are categorised into:

- *Form-based representations*, including *menus* and *tables* as specialisations of forms (e.g., [171,172]), adopt conventional paper forms as a metaphor.
- *Icon-based representations* employ images or symbols, representing abstract and real objects by exploiting analogy, convention, etc., to serve domain knowledge and application functionality.
- *Diagram-based representations* utilise geometric symbols and a spatial layout to depict relationships among schema concepts.
- *Hybrid representations* are based on combination of several visual paradigms.

Regarding the interaction paradigm, it is viewed in twofold with respect to understanding the reality of interest and query construction. For the former, three categories are considered:

- *The top-down technique* breaks the concepts in a schema into a set of layers, with respect to a certain criteria (e.g., level of importance - *selective zoom*, hierarchy - *hierarchical zoom*); each layer provides access to preceding and following layers.
- *The browsing technique* allows traversing the concepts of a schema and/or data (i.e., *intension* and *extension*), by exploiting the relationships between concepts.
- *The schema simplification technique* aims to generate simpler user views, by aggregating and transforming the concepts of a schema.

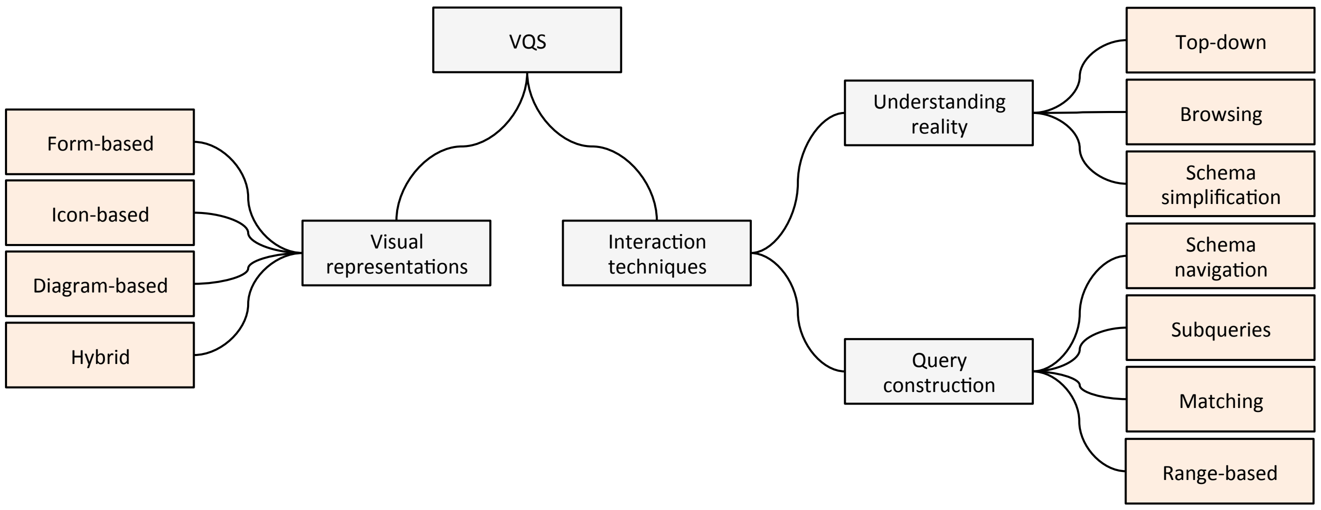


Fig. 2 The classification of VQSs with respect to representation and interaction paradigms.

For the query construction, the following categories are suggested:

– *The schema navigation technique* lets users to construct a query by starting from a central concept and following the paths between concepts for including other concepts of interest.
– *The subqueries technique* allows composing concepts or partial results provided by subqueries to realise query construction.
– *The matching technique* lets users to provide the structure of a possible answer through an example or pattern to be checked against data.
– *The range-based technique* allows users to manipulate a set of *input widgets*, such as sliders and button to instantly filter data.

Finally, note that query formulation is an *iterative process*, that is, a user explores the conceptual space, formulates a query, inspects the results, and reformulates the query until she/he reaches to the desired query, and each iteration could be considered an *attempt*.

3.2 On user types

The potential users of a visual query formulation tool could come from different backgrounds and have varying levels of expertise. Domain expertise (i.e., knowledge on a specific subject matter such as medicine, fishery, and oil exploration) and technical expertise (i.e., technical knowledge and skills on design, programming, and modelling software artefacts) could be used to differentiate between different types of users.

Catarci et al. [37] distinguish between *professional users* and *non-professional users*. The former refers to users who possess a wide spectrum of skills on programming languages, database management systems, etc. The latter refers to users who cannot invest time in computer training and usually learn query languages by doing and are further classified into *familiar* and *unfamiliar* users with respect to familiarity with the database semantic domain.

Dadzie et al. [51] considers two main types of users, which are *lay/mainstream users*, and *tech users*. Users of the first category are computer literate and can find information online, and generally do not have in-depth domain knowledge. Users in the second category understand the Semantic Web and other advanced technologies, can use RDF, and understand an ontology. A third sub-category is proposed, namely *domain experts* who have in-depth domain knowledge to interpret and understand relevant data, and might have knowledge on the Semantic Web technologies.

The categorisation provided by Dadzie et al. [51] could be problematic, since a programmer without expertise in the Semantic Web technologies could be considered as a lay user, since the definition exclusively focuses on expertise in Semantic Web. However, users with substantial technical background, regardless of their expertise in the subject textual language, ontology language, and technology, possess semantic knowledge on programming languages, systems, frameworks, and tools, which is non-volatile and easily transferable while using a new one (cf. syntactic/semantic model [150]).

In this article, users are considered in three categories, *casual users*, *domain experts*, and *IT experts*. Casual users use computers in their daily life/work for basic tasks, such as typing documents, e-mails, and web browsing, without any substantial knowledge in the domain of interest. They have low tolerance on formal languages and are unfamiliar with the technical details of an information system (cf. [50]). Domain experts have an in-depth knowledge and an understanding of the semantics of their expertise domain, while IT experts have technical knowledge and skills on a wide spectrum of topics, such as programming and databases. This categorisation is neither complete nor mutually exclusive e.g., a domain expert could be also an IT expert.

Casual users and domain experts with no technical background could benefit the most from visual query formulation and the ultimate benefit of it questionable for IT experts, since in many cases they might find working on a textual language more efficient and non-limiting (cf. [37,150]). Therefore, casual users and domain experts (without technical expertise) are the focus of this article, and are further classified into a parent category, *end users*. End users cannot/do not desire to use textual languages to retrieve data due to the lack of technical knowledge and skills and might have domain knowledge on the subject matter. Visual query formulation is an *end-user development* paradigm, which is defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, to create, modify, or extend software artefacts [112]. This very paradigm underlies the definition of end user in this context.

3.3 Evaluation: usability and expressiveness

Usability and *expressiveness* are the two interplaying dimensions of visual query formulation. The usability (cf. [21]) of a tool reflects whether it is competent of meeting its identified aim, while expressiveness (cf. [37, 61]) defines the ability and breadth of a tool to characterise the domain knowledge and information needs.

Usability is characterised in terms of *effectiveness*, *efficiency*, and *user satisfaction*. Effectiveness (cf. [36, 21]) measures the *accuracy* and *completeness* that users can achieve (i.e., "doing the right things"). Efficiency (cf. [36,21]) reflects the cost associated with the level of effectiveness achieved, and is mostly measured in terms of the time spent to complete a query (i.e., "doing the things right"). User satisfaction (cf. [113]) refers to the perceived quality of dialog and user interface and plays a determining role in the attitudes of users, such as *trust*, *engagement*, *acceptance*, and *comfort*, against the tool. It usually has considerable potential to cause failure, if not taken seriously. User satisfaction is usually measured by questioning the attitude of users after experiencing with the subject system or language through surveys, interviews etc. (e.g., [113,42])

Note that, typically, in IR systems effectiveness is measured in terms of *precision*, *recall*, and *f-measure* (harmonic mean of precision and recall) over the result set; however, as stated earlier, for a DR system, a single missing or irrelevant object implies failure. Therefore, for a VQS and VQL, effectiveness is rather measured in terms of a binary measure of success (i.e., correct/incorrect query), which could be accompanied with a fuzzy measure (i.e., the rate of accuracy and completeness) weighting and combining different types of query errors (cf. [185,94]). As far as SPARQL is concerned, the measure of correctness could be built on the semantic similarity between the user query and correct query (cf. [55]). Since query formulation is an iterative process, allowing and incorporating multiple attempts into account is also a sensible approach.

User studies are typically realised by means of *query writing* and *query reading* tasks (cf. [37,168]) to evaluate the tool itself alone or to compare it with others either with a *summative* or *formative* perspective; various other studies and measures (e.g., *learnability* [128]) can be defined as well. However, consulting users only at the end (i.e., summative evaluation) does not help much (cf. [89,36]). The active participation of users at every cycle of development, grounded on a *user-centred design* [130] approach with intermediary reflective assessments (i.e., formative evaluation), is the true contributor.

Wilson et al. [182] employ evaluation as a part of the overall design process to inform the design refinements and argue that, particularity in comparative evaluation, taking user interface as a sole independent variable would reveal very little about why one outperforms another. In this respect, the authors propose a formative evaluation framework by using three established *information-seeking models*, namely *stratified*, *episodic*, and *strategic* models, among others in the literature (cf. [182]). The first model forms the basis of evaluation framework in the form of abstraction levels used to evaluate the design, while the latter two correspond to these abstraction levels. The episodic model defines a set of information-seeking scenarios and the strategic model defines different *tactics* for interacting with the information, which are then mapped into the episodic model. The evaluation framework includes the identification of *features* (e.g., filtering, grouping, keyword search) in each subject interface with the number of *moves* (e.g., scroll and select) required to realise each tactic with each feature. The results are used to find and compare the amount of user effort required to realise each tactic, feature, and search scenario with each interface.

Regarding the expressiveness of a VQSs and VQLs, a formal evaluation is required. However, the formal evaluation of a VQS is not as straight forward as of a

VQL. A VQL has clear formal semantics that allow a rigorous computational evaluation (cf. [155,66]), while on the contrary, a VQS is built on an arbitrary set of user actions that effectively capture a set of syntactic rules specifying a (query) language. Thus, the formal evaluation of a VQS is a considerably harder problem. Indeed, for didactic purposes, one could establish an analogy with *Protégé* [99], which is an interactive system for ontology editing. The assessment of Protégé's ability to cover all ontology constructs remains ad-hoc, while the expressiveness of the underlying ontology language, such as *Web Ontology Language (OWL)* (cf. [166]), has a clear formal grounding. A formal framework for evaluating the expressiveness of VQSs could borrow techniques and approaches from *formal software verification* domain. Particularly static approaches for interactive systems (e.g., [53,35]) could allow the construction of mathematical models for user actions and system behaviour.

In most cases, a VQS is intentionally kept less expressive than the underlying language due to the expressiveness and usability trade-off (cf. [168]). Complex semantics expressed with visual constructs and operations may result in inefficiency for IT experts and difficulty in use for *end users* (cf. [150,37]). Therefore, expressiveness is quite often sacrificed for the sake of usability.

4 Ontology-based data access

The fact that today relational databases hold a significant amount of the world's enterprise data is a strong barrier against the adoption of ontology-based visual query formulation approaches. However, the emergence of OBDA technologies (cf. [164,101]) raised ontology-based visual query formulation as a viable and promising approach for querying a variety of structured data sources.

OBDA built on *data virtualisation* enables *in-place* querying of legacy relational data sources over ontologies. That is data stays where it is and in its original structure, and an ontology comes in as an additional layer with appropriate mechanisms to virtualise underlying data into RDF. As stressed by Kogalovsky [101], with the OBDA approach, a traditional database system is transformed into a knowledge base, with a multilayer architecture coherent with the data independence principle (cf. [59]). The database system is employed for the assertional knowledge (called *Abox* in DL) and query evaluation (i.e., retrieval of stored facts), and ontology is employed for the terminological knowledge (called *Tbox* in DL) and *query answering* (i.e., retrieval of implicit and explicit facts). There already exist various OBDA frameworks and systems such as *OntoQF* [126], *Mastro* [44], *Ontop* [143], *Ultrawrap* [149] and *Morph* [138].

OBDA systems fall into the category of *ontology-driven information systems* (cf. [75,145]), under which ontologies are used as external run-time artefacts. However, the marriage of ontologies and database systems is built on the long-standing legacy of a plethora of interrelated approaches aiming at improving the semantic level of databases (cf. [101,120]) grounded on the *relational model* [46]. On the one hand, *object-oriented* approaches for database systems, such as *object-oriented databases*, *object-relational databases* and *object-relational mappings* (cf. [59,133]), emerged to bring the representation capabilities of databases closer to the richer object-oriented applications (e.g., type system, class hierarchy). On the other hand, approaches dealing with the integration of LP and database systems (cf. [48]), for the sake of combining formal reasoning with the capability of handling large amounts of data, led to *deductive databases*; a prominent example is *Datolog language* [123] and its extensions (e.g., [33]). This has been followed by a hybrid approach, referred to as *object-deductive databases*, which combines the representation capabilities of object-oriented databases and the reasoning power of deductive databases; *F-logic* [98] is a well-known example.

Despite the relative success of deductive and object-deductive databases, approaches based on *description logic* (DL) hold a leading and active role in OBDA (cf. [101]). DL is a family of knowledge representation languages and aims to reach a profitable compromise between expressiveness and reasoning by weaving logic-based semantics over structured knowledge. The activities and standards of the *Semantic Web*, particularly with the OWL and its profiles that are built on DL, foster this line of research (cf. [17,22]). Notably, *OWL 2 QL*, which is based on *DL-Lite*, is particularly meant for applications for query answering with a large amount of instance data and in this profile query answering can be implemented by rewriting queries into a standard relational query language [124,73].

4.1 Techniques and practice

A prominent direction for integrating DL ontologies and databases is founded on *mappings* and *query rewriting* (cf. [101,164,144]) – see Figure 3. The approach is backward compatible, preserves existing traditional databases (e.g., SQL databases) as data sources, and utilises ontologies as external higher-level structures (i.e., super structure) over the database schemas (cf. [69]). This is realised through a set of mappings, which describe the relationships between the terms in the ontology and their representations in the data sources (i.e., data and database schema).

The approach is analogous to the object-relational mappings, where two common techniques exist. In the first technique, a mapping between domain concepts and database schema is provided (e.g., *Relational Persistence for Java and .NET - Hibernate*[1]); it provides high transparency, though is difficult to realise for complex cases. In the second technique, the domain concepts are linked to the parameters and results of SQL queries (e.g., *iBatis*[2]); it is easier but requires more effort and a good understanding of database schema and system. The second technique is employed by many OBDA solutions (e.g., [144,142]), while recent examples for the first technique are also available (e.g., [126]).

In any case, queries are formulated with ontological terms (e.g., in SPARQL); and, once submitted, they are transformed. Two rewrites take place in this context (cf. [143]); in the first one, the query is rewritten by taking ontology constraints into account. That is it uses Tbox axioms and unification to generate more specific queries from the original, most general query [142]. This is because automated reasoning could be computationally very expensive and most standard reasoning techniques would need to interleave operations on the ontology and the data, which may not be practically feasible if the data is stored in a relational database [69]. In the second rewrite, the query is transformed into the language of the underlying relational database system (e.g., SQL) by the OBDA framework through available mappings. At this stage the query could be further optimised in order to improve execution performance (cf., [141]). A highly simplified example is given in Figure 3; for more technical information on existing technologies, interested readers are referred to [101,164, 126,144,142,141].

A key benefit of the mapping and rewriting approach is avoiding the representation controversy (cf. [120]) between ontologies and relational databases (i.e., *impedance mismatch*), centred on their semantic and syntactic differences (cf. [157]), by separating transactional and domain perspectives. That is, while exploiting ontologies for data access and reasoning, one could also enjoy the benefits of well-established query optimisation and evaluation support available for traditional database systems without migrating or duplicating any data (i.e., *materialisation* [164]). Another benefit is *federation*, that is the ability to integrate distributed data sources with differing schemas by relating each to a common ontology. Also, the use of the Semantic Web technologies and standards makes it possible to apply and integrate the approach into a broader context, such as for public data available on the Web (e.g., *linked data* [22]) and for *semantic interoperability* (e.g., semantic *service mashups* and *data mashups* [174,163]). From a wider perspective, the approach also opens up the possibility of automatically deriving database schemas and mappings, even the portion of application code, from the subject ontologies (e.g., [157]). This relates to the design time use of ontologies, namely *ontology-driven design of information systems* (cf. [75,145]).

[1] http://www.hibernate.org

[2] http://www.mybatis.org

5 Challenges and Requirements

Expressiveness and usability span the challenges and requirements for a visual query formulation tool. One can reach a reified sphere of analysis by projecting main data access activities, exploration and construction (cf. [37]), into this frame – see Figure 4. For each activity, the qualities of *presentation* and *interaction* are of focus as employed visual representation and interaction paradigms and how they are put together, where alternatives are widely available, are of importance. Here, the notion of presentation not only concerns the representation of domain knowledge, but rather every possible graphical aspect of a user interface; and, interaction refers to a two-way effect (i.e., *control-feedback loop*) between a user and an application, which could be addressed in terms of *control* (i.e., of user) and *behaviour* (i.e., of application). In the following, general requirements are discussed in terms of *graphic* and *interaction design* perspectives, and in subsections more specific challenges and requirements are discussed.

Likewise, in this context, the expressiveness and usability are bi-directional in nature. One is from a system or language to user point of view, pertaining to the *ability*, *extent* and *quality* of a language or system for representing domain knowledge and elicited information needs. For instance, one might be concerned whether a system or language is able to visually represent *subsumption* and *disjointness* between domain concepts and at what level of quality. The second one is from a user to system or language point of view, pertaining to the ability, extent, and quality of a language or system for enabling users to articulate their information needs. For example, one might be concerned whether a system or language allows users to visually express *conjunctive* and *disjunctive queries* and at what level of quality.

Concerning the expressiveness, primarily, a presentation should encode all relevant information [116]), pertaining to domain and information needs, and only that information. However, the expressiveness of an interface not only relies on static visualisations, but also

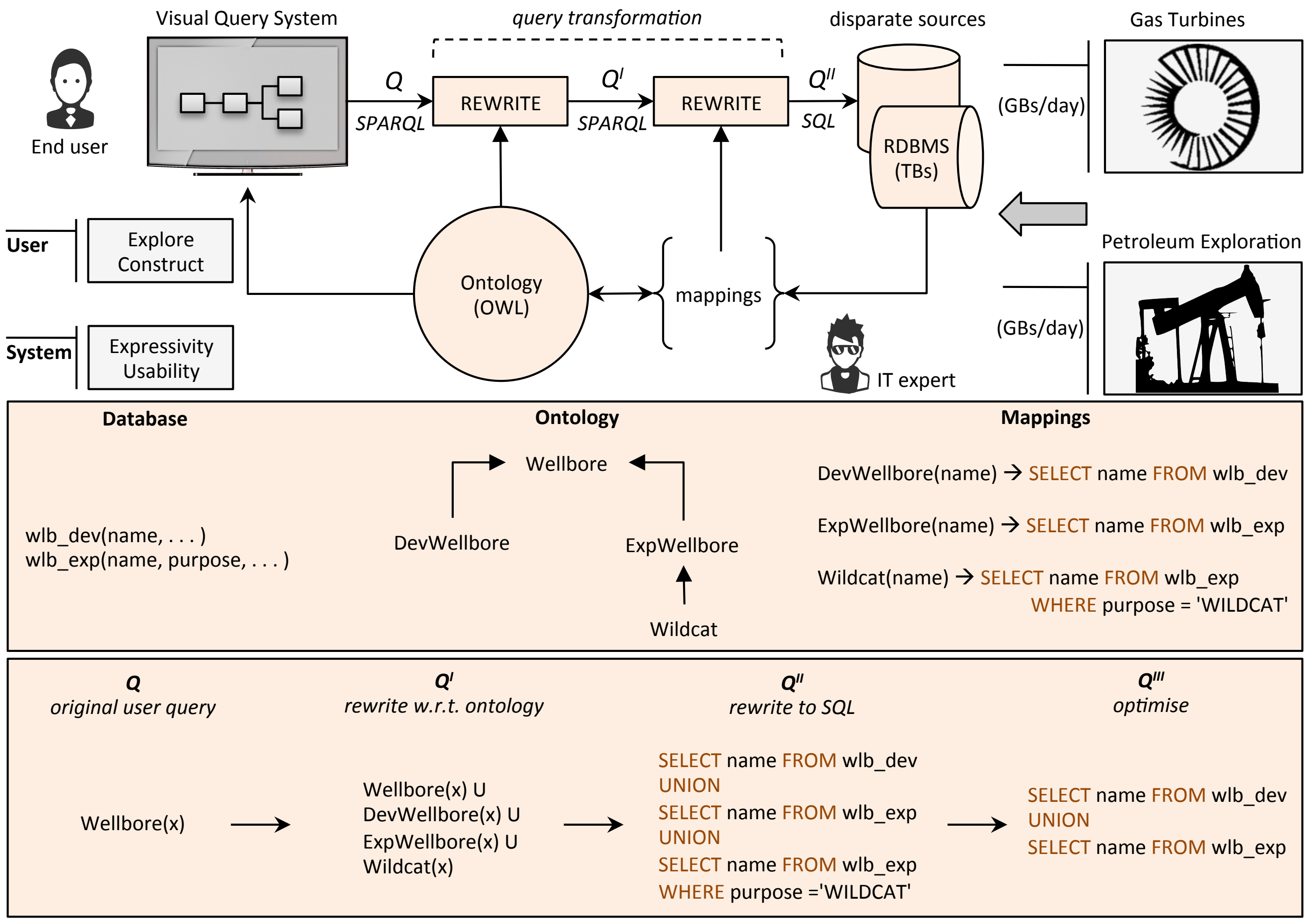

Visual Query System
query transformation
disparate sources
Gas Turbines
End user
Q
SPARQL
REWRITE
Q'
SPARQL
REWRITE
Q''
SQL
RDBMS (TBs)
(GBs/day)
Petroleum Exploration
(GBs/day)
User
Explore Construct
System
Expressivity Usability
Ontology (OWL)
mappings
IT expert
Database
wlb_dev(name, . . .)
wlb_exp(name, purpose, . . .)
Ontology
Wellbore
DevWellbore
ExpWellbore
Wildcat
Mappings
DevWellbore(name) → SELECT name FROM wlb_dev
ExpWellbore(name) → SELECT name FROM wlb_exp
Wildcat(name) → SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'
Q original user query
Q' rewrite w.r.t. ontology
Q'' rewrite to SQL
Q''' optimise
Wellbore(x)
Wellbore(x) U DevWellbore(x) U ExpWellbore(x) U Wildcat(x)
SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp UNION SELECT name FROM wlb_exp WHERE purpose ='WILDCAT'
SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp

Fig. 3 Visual query formulation and ontology-based data access.

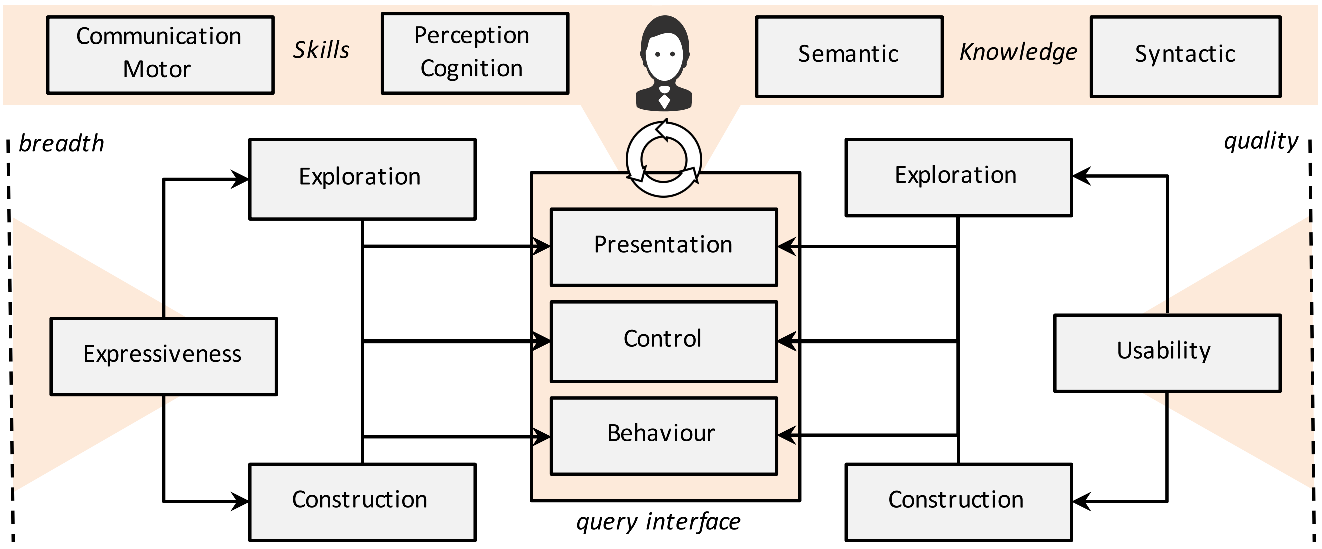


Fig. 4 A basic requirements framework for visual query formulation.

on interaction. A query interface should provide necessary affordances and means to explore domain and to construct and manipulate queries, under user direction, over time (cf. *potential expressiveness* [18]). Yet, user control commands and application behaviour have to be restricted to the constraints of domain, for instance, no user action or application behaviour should lead to a state that overlaps two disjoint concepts. Generally, the need and the borders of user skills and knowledge define the confines of expressivity, whilst the latter being a highly determining factor.

The usability of an interface concerns the way it utilises human natural capabilities, particularly *communication*, *motor*, *perception*, and *cognitive* skills (cf. [176]), and knowledge (cf. syntactic and semantic knowledge [150]). The first two relate to user control, while the last two pertain to the visual and semantic aspects of presentation and application behaviour. The ground challenge is to provide familiar metaphors and interaction styles so as to increase the magnitude of preconscious processing (e.g., recognition vs. recall) and to foster innate user reactions (e.g., clicking vs. typing). These are built on natural visual elements and cues and are spanned by common human skills and knowledge, for representing domain knowledge, application behaviour, and constructive and manipulative functionalities.

Apart from traditional design considerations, one needs to deal with *scalability* problems that come with Big Data. Big Data refers to the collection of data sets characterised by high *volume*, *velocity*, *variety*, and *complexity* (cf. [107,69]). These four dimensions do not only concern data but also schemata; therefore, one should consider scalability both in terms of query evaluation and query formulation – see Figure 5). The former concerns the cost of executing queries against data sources (cf. [117]), while the latter refers to the user effort for finding and using relevant elements of ontology to form the desired query. In this article, the query formulation is of concern and the scalability issues could be better described in relation with the dimensions of Big Data:

- Complexity: ontologies with varied and sophisticated relationships and constraints (e.g., a deep and large class hierarchy) are harder to represent, comprehend, and construct queries from (cf. [93,69]).
- Volume: large ontologies are hard to visualise and explore on a finite display, and a large data size makes it difficult to spontaneously interact with the underlying data (cf. [68,118,93]).
- Variety: ontologies spanning varying types of data and data sources are harder to represent and interact with generic representation and interaction paradigms (cf. [173,57]).
- Velocity: rapidly changing data sources (e.g., stream) are difficult to exploit with passive visual query formulation tools (i.e., non-proactive and non-reactive) (cf. [183,69]).

5.1 Expressiveness

An ontology construct, feature, or functionality provided by a VQS or VQL makes value only if users need, understand, and use it. Hence, it is appropriate to approach expressiveness from a user perspective (cf. [36]). That is, *a*) a visual query formulation tool should primarily span the types of information needs and ontology constructs needed by users in their context; and *b*) the inclusion criteria should be the *user perceived complexity* (i.e., the accessibility of a construct, feature, or functionality) rather than the formal aspects of the underlying formality.

The user perceived complexity is a crucial factor in determining the confines of expressiveness. Therefore, one should be aware that a visual query formulation tool aimed for end users is not expected to have one-to-one correspondence with the underlying textual language (i.e., full expressivity) or be rigidly formal and comprehensive like ontology editors and visualisation tools used for *ontology design and management* (e.g., [106,93]).

5.1.1 Exploration

Exploration is indeed a *continuous* activity, since query formulation demands users to actively engage with the domain knowledge and semantics; an inadequate supply of such information leaves users confused or in vacuum. A user may either purely explore to gain some insights with/without a concrete formulation task in mind or to seek for more knowledge for the next step at any point of an active task.

Katifori et al. [93], in their survey, provide a comparative analysis of ontology visualisation methods with respect to their ability, efficiency, and effectiveness to represent *class taxonomy* (i.e., class/subclass hierarchy), *multiple inheritance*, *instances*, *attributes*, and *relationships*. However, among others in OWL, *enumerated classes*, *disjointness*, *subproperties*, *inverse properties*, and *multiple ranges* are also important to aid users. Remaining constructs are not deemed to be essential (i.e., of OWL 2 [166]), such as *role chains* and *transitivity*, since they are mostly valuable at query answering stage in terms of *classification*, *inference*, and *consistency checking*.

For instance, on the one hand, consider a transitive property “subRegionOf” for representing part-whole

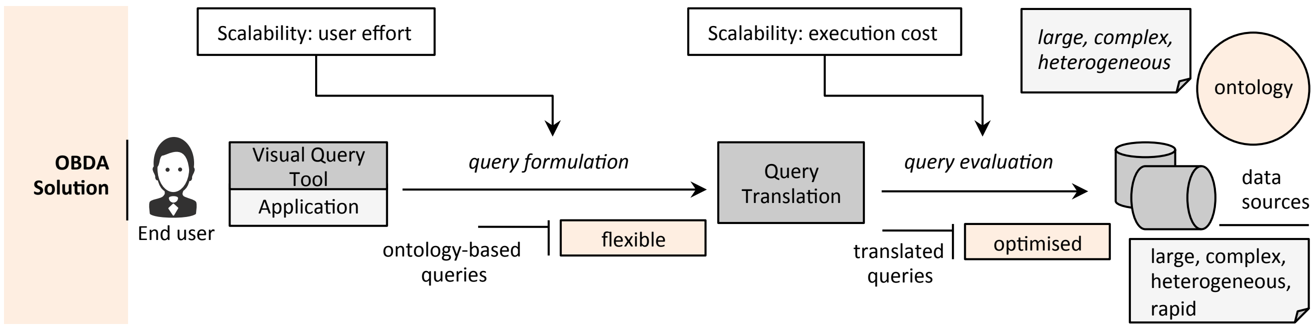


Fig. 5 Visual query formulation, OBDA, and scalability issues.

relationships between geographical regions. Knowing about the transitivity of this property is not of any help for users in constructing queries for retrieving the subregions of a city, since users are interested in the intended meaning, not in how it is realised. On the other hand, consider two disjoint classes or an enumerated class. The information on the nature of these classes aids users in selecting the class types and values effectively, by not allowing the selection of two disjoint classes or of an inappropriate value respectively. Nonetheless, multiple inheritance, disjointness, subproperties, inverse properties, and multiple ranges are comparatively harder to communicate, while others are mostly well established in many ontology visualisation methods and tools (cf. [93]).

Ontologies have a *propagative effect* on the amount of information to be presented. This case is considered in two forms, namely the *top-down* and *bottom-up propagation of property restrictions*. The first form emerges from the fact that, in an ontology, explicit restrictions attached to a concept are inherited by its subconcepts. For instance, consider an ontology that includes a superconcept "Person", its subconcepts "Lecturer" and "Student", and a data type property "hasID" with domain "Person". It may be of use to explicitly present "hasID" property, while viewing the subconcepts of "Person", although the connection is not explicit. The second form is rooted from the fact that the interpretation of an OWL concept also includes the interpretations of all its subconcepts. Therefore, for a given concept, it may also make sense to suggest the (potential) restrictions of its subconcepts. For instance, for the same ontology, assume an object type property "teaches" whose domain is "Lecturer". It may be of use to present "teaches" property, while viewing the superconcepts of "Lecturer". Yet, largely due to the lack of provenance and the destruction of hierarchical view, such information, albeit useful, is harder to represent.

An often employed mechanism for exploring ontologies is navigation approach. An ontology could be considered as a *graph*, whose *nodes* are mainly hierarchically organised concepts and *edges* are relationships. However, in many cases, an ontology is more than a network of hierarchies and it is not always straightforward to represent an ontology in terms of a graph. For instance, 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. In this respect, a technique for extracting graph-like structures from OWL 2 ontologies is employed by Soylu et al. [161]. To realise the idea of ontology guided navigation, the visual query formulation tool needs to 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, it is ensured that the tool mimics the structure of (and implicit information in) the ontology and data and that the interface does not contain irrelevant (combinations of) elements.

5.1.2 Construction

A query basically describes a subgraph of an ontology. Such descriptions are typically composed through *select*, *join* and *projection* functions (cf. [59]). The select operation is meant for specifying conditions on concepts and on their mergers for filtering down the result set (e.g., *where clause* in SQL), while the project operation determines attributes that are to be displayed (i.e., *select clause* in SQL). The join operation represents the traversal of adjacent elements through binary relationships to relate two or more concepts. The select and projection operations are comparatively easier, while the join operation is problematic for many users (cf. [89]). Although ontology-based approaches introduce a more natural way of joining concepts, there are still issues, which are discussed below.

A fundamental distinction between queries depends on how query conditions are connected and is considered in twofold: *conjunctive* and *disjunctive queries*. The former are basically realised by merging query conditions with logical "AND" connectives (cf. [40,71]), while the latter are formed with logical “OR” connectives (cf. [45]). Queries can be seen in several topological categories, such as *linear queries*, *queries with branching* and *cyclic queries*. The first refers to queries that are basically serial joins of concepts, while the second category of queries are composed of *path expressions* with *branching* (cf. [7]). The latter refers to queries that have at least a path (undirected) in the query graph whose first node corresponds to the last. An example cyclic query would be “find all persons who work in the city that they were born” (“bornIn” and “worksIn” are relationships, while “person” and “city” are concepts). Queries including disjunction and cycles are more problematic compared to linear and *tree-shaped* conjunctive queries and cycles are usually addressed through *node duplication* approach (cf. [36,93]) – i.e., by duplicating the city node in the previous example.

There are query types that cannot purely fit into the aforementioned topological forms, such as queries with *quantification*, *negation*, and *aggregation*. Two fundamental forms of quantification are *existential quantification* and *universal quantification* (cf. [126,95]). Existential quantification is interpreted as “there exists”, “there exists at least one”, “for some”, and in OWL “some values from” (cf. [10,166]); an example query would be “find all lecturers who teach at least one math course”. Universal quantification is interpreted as “for all”, “for every”, “only” and, in OWL “all values from” (cf. [166]); an example query would be “find all lecturers who only teach Introduction to Computing course”. Negation is an operation that may be applied on a *proposition*, *truth-value* etc., and in the simplest terms, is used to reverse a condition; an example query would be “find all students who do not take Introduction to Computing course” (e.g., [10]). Aggregation functions such as *count*, *sum* and *max* are used to group values of multiple attributes to form a single value. While the use of existential quantification remains implicit, queries that include universal quantification, negation, and aggregation are quite esoteric for end users; this even applies to skilled users, particularly for universal quantifiers (cf. [95]).

Finally, *query manipulation* and *query reformulation* are often expected situations. However, query manipulation, as opposed to query reformulation from scratch, needs special attention. This is due to the fact that while revising a part of an existing query, the overall semantic consistency has to be maintained (cf. [7]). For instance, assume that a user wants to specialise a relationship; this change does not have any global effect. However, on the contrary, assume that the user wants to alter a relationship to one of its siblings, where its sibling and the current relationship range on two disjoint classes respectively or the user wants to specialise one of the classes. Such cases require the remaining of the query to be altered as well. Another example could be the deletion of a node from the middle of a path expression; in such a situation, the rest of the path could be deleted automatically. For the convenience of the user, however, a better approach would be suggesting new connections, which is not always straight forward, and a delete option. The consequences of the query manipulation have to be taken care of by the query formulation tool rather than the user, yet the semantics and reasoning behind the subsequent changes done by the interface could be hard to communicate and explain to the user.

5.1.3 Discussion

The user perceived complexity of above-mentioned ontology constructs and query types ranges with respect to representation and interaction paradigms, for instance, while list/menu-based approaches are good for representing class taxonomies, they are not well suited for representing relationships (cf. [37,93]). One should also see that, a system-based approach (i.e., VQS) could provide more possibilities in terms of expressivity than a language-based approach (i.e. VQL), since there are no rigid formal boundaries for a VQS. For instance, in a VQS, aggregation functions could be handled with tabular data view, which is actually meant for result display.

The amount of information a paradigm can communicate, the amount of information a user is able to discern from it (i.e., *perceptual level*), and the amount of load a paradigm places over the user (i.e., *cognitive level*) are among the selection criteria. Therefore, one could rightly assert that expressivity should be considered at perceptual and cognitive levels. Nonetheless, one should also realise that ontology and the query that is reflected back, in many cases, need to be oversimplified for end users. Database/IT experts, given the complexity of an ontology, might consider this ambiguous; however, the reference point should be the end users.

End-user visual query formulation comes with a considerable gap between tasks and users in terms of required and possessed competence. However, simpler visual interfaces are expected to suffice for the majority of end-user queries. 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 (cf. [36,110]). A complex interface is likely to be rejected by the end users; however, some end users might be more advanced with respect to others. A possible approach could be dividing the functionality into layers, where a user could start using a system with minimal functionality and unlock complex functionalities as his/her competence progresses (cf. [150]).

Indeed, there is no infinite usability; therefore, the main goal should be to address frequently occurring query formulation tasks that are reasonably complex. Therefore, a visual query formulation tool is often limited in expressiveness with respect to the underlying formality; however, the expressiveness of the underlying language is important (e.g., *relational completeness* [37,61,47,5]), since queries that cannot be addressed at the end-user level are likely to be delegated to users who directly work with the target textual language. A complex query construct should either be completely discarded and delegated to users with IT skills, if it is found to be very complex for end users. Alternatively, its simple/restricted forms should be supported and complex forms should be delegated to skilled users – e.g., a purely disjunctive or conjunctive set of path expressions vs. a nested set of path expressions connected with disjunctions and conjunctions.

It is important to gather a catalogue of example queries categorised into a set of prototypical classes. This is of help for finding an appropriate compromise between the need and complexity, while selecting the representation and interaction paradigms. In this article, queries are categorised into three levels according to the need and complexity. The *first level*, *ground queries* [168], refers to simple linear and tree-shaped conjunctive queries, while the *second level* refers to queries with disjunctions, aggregation and cycles. The *third level* refers to queries with universal quantifiers, negation, and query manipulation support. It is postulated that, in many cases, the majority of end-user queries are to be centred around the first level. This categorisation largely relies on personal experience and insights gained from the literature; a rigid classification could be performed through end-user studies.

The complexity does not purely originate from the inherent difficulty of ontology and query constructs; it is also bound to the size. For instance, Catarci [36] categorises queries into *close-acyclic*, *far-acyclic*, *far-cyclic*, *very far-acyclic* queries, in which not only the structural form, but also the lengths of queries matter. Yet, at this point, the matter becomes largely a usability issue, presented in what follows, for which the representation and interaction paradigms should be tailored to provide gradual access and construction mechanisms for large ontologies and queries.

5.2 Usability

The usability of a visual query formulation tool concerns the selection and intertwining of representation metaphors, visual attributes, and interaction styles that require less knowledge, skills and learning effort, and allow users to *discern*, *comprehend*, and *communicate* a maximum amount of information effortlessly (cf. [140]). In general, in order to meet these goals, the tool should be *intelligible*, *intuitive*, *succinct*, and *stimulating*, while providing *instant*, *gradual*, *iterative*, and *reversible* experiences. It should also be integrated and adapted to *context* (cf. [156,49]), such as *personal*, *data-related*, *task-related*, *organisational*, and *environmental*.

A visual query formulation tool needs to provide a wide support ranging from situating and orienting users in the conceptual space to helping users in understanding and using data. However, *the Big Data effect* impedes the use of any visual query tool. Primarily, volume and complexity hinder human perception and cognition respectively, while variety and velocity require going beyond generic representation and interaction paradigms and on-demand querying approaches.

5.2.1 Exploration

Exploration is not only a continuous but also an *omnipresent* activity. At any phase of a user experience, the display medium presents a portion of the domain, which basically means either an *active* or *passive exploration*, where the user intention is mainly explorative or constructive respectively. For this reason, ontology visualisation is an integral part of query formulation.

Katifori et al. [93] categorise ontology visualisation methods into *indented list*, *node–link and tree*, *zoomable*, *space-filling*, *focus + context or distortion*, and *3D information landscapes* with a hybrid perspective of interaction style (for understanding the reality of interest) and representation paradigm (cf. the classification of Catarci [37]). Indented list approaches represent the taxonomy of an ontology as a tree (cf. menu/list-based approaches), while in node–link and tree approaches, a set of interconnected nodes represent the taxonomy and properties of an ontology (cf. diagram-based approaches). Zoomable approaches present the nodes on lower levels of hierarchy nested inside their parents, and zooming changes the viewing level (cf. top-down interaction style). The space-filling approaches use the screen as a whole by subdividing the space available for a node among its children and each subdivision corresponds to a property of the node assigned to it (cf. diagram-based approaches). The focus + context or distortion is based on the notion of distorting the view of the presented graph in order to combine context and focus, where the node on focus is usually the central one and the rest of the nodes are presented around it (cf. schema simplification). In 3D information landscapes, nodes are placed on a plane as colour-coded and size-coded 3D objects (cf. diagram-based approaches).

As both Katifori [93] and Catarci [37] suggest each representation paradigm has its advantages and disadvantages, for instance, while indented lists are better at representing hierarchies, node–link trees are better at representing relationships between concepts. However, with any interface, there are technical limits on what can be presented on a finite display space and perceptual limits on how much the user can take from the visual representation (cf. [177,93]). Here, the first challenge is to diminish the effect of increasing volume and complexity of domain knowledge to prevent users being lost in the conceptual space.

Exploration with large numbers of concepts, relationships, and attributes in a highly complex domain is a hard problem, since the presentation could easily become overcrowded and cluttered and prevent users to reach an overall understanding or to find particular information (cf. [171,93,137]). In query formulation, the volume and complexity largely matter in terms of schemata rather than the data, since users interact with the system primarily at a conceptual level. The challenge is to mitigate generating large and incomprehensible visualisations (cf. [137]). The very first approach to scale against the complexity and volume is to provide intuitive interaction styles that allow users to gradually explore and access domain knowledge.

Shneiderman [152] defines seven tasks for visual information access, namely, *overview*, *zoom*, *filter*, *details-on-demand*, *relate*, *history*, and *extract*. This categorisation is of help for classifying required interaction types. The overview and zoom tasks are of use for forming a global understanding and changing vertical focus within the conceptual space respectively (cf. top-down). The zooming task is not meant to be a geometrical rescaling, rather it allows focusing on a specific part of the presentation, while increasing the depth and the magnitude of detail (cf. semantic zooming [135]). A complementary task could be *viewpoint movement*, which changes the focus horizontally to another part of presentation (cf. [93]). Filter and details-on-demand tasks are meant to isolate a fragment of knowledge that is of interest. The relate task enables users to roam the conceptual space by pursuing the relationships among nodes (cf. browsing). This could be realised through expansion and retraction of nodes, which results in an orderly change in the viewpoint. The history task supports users to control and reflect on their previous actions (e.g., undo, replay, develop experience), while extract task corresponds to query construction actions and is discussed in the following subsection.

Schema clustering and *summarisation* are two concrete means to increase comprehensibility and communicability. Schema clustering approaches aim at automatically adding abstraction layers to conceptual schemas (e.g., [34]), after which users are not first confronted with hundreds or thousands of concepts, rather with high-level clusters that they could drill down. Schema summarisation is meant to provide a visual overview of the entire domain to aid user understanding (e.g., [110, 186]). A simple *meta-level visualisation*, for instance, could reflect the number of instances associated with each concept and the number of relationships between concepts by using variable size thickness for shapes and lines representing concepts and relationships respectively. More advanced visualisations are possible through *network visualisation* techniques that can handle high numbers of concepts and use different pattern identification approaches (cf. [108]).

Another promising direction is to employ *adaptivity* and *customisation* (i.e., adaptability) techniques (cf. [29, 146]). Adaptivity allows a system to automatically alter/adapt its *content*, *behaviour*, and *presentation* with respect to changing context dimensions, where customisation is a user managed process of altering software to his/her particular context such as needs and preferences. Context is a broad notion that encompasses any information that characterises the computing context including entities involved (e.g., user, devices, time, location) and relationships among them (cf. [54,20]). Although the potential of adaptivity and customisation for enriching the usability of visual query formulation tools is already apparent, it is not yet truly exploited (cf. [36, 89]). An example could be the physical environment, if a user is working on two or more screens, the part of the visualisation can be delegated to other display units (cf. [78]). Another example could be the device, if the user is on the field and accessing the tool from a mobile device, the tool could alter its presentation to the screen characteristics of the device and provide a more natural interaction mechanism, such as *haptic control* [109], which complements the direct manipulation approach.

The high variety and velocity also have implications on visual query tools at the exploration stage. Specific representations that suit best to the nature of data at hand, along with generic presentation facilities, are required to better communicate and interact with different types of data. This may include specific presentations for data fragments ranging from atomic facts, such as time and location, to high-level concepts, such as people and places, which may also include atomic facts (cf. [173]). Two very prominent examples are *spatial* and *temporal* data sources, where the use of domain-specific presentations is reported to be more effective (e.g., [57,39]).

The presentation elements used in such scenarios are mostly analogous to real-life, such as maps, which lead to immediate grasping by increasing the magnitude of *preconscious processing* (cf. [65]).

5.2.2 Construction

The transition between exploration and construction is mostly obscure and needs special attention due to their adverse, though complementary roles. The goal of a user at the query construction stage is to travel all the paths that form the intended query as quickly as possible, unlike the exploration stage where the aim is to discover the conceptual space to a large extent. Hence, for the sake of exploration, visual query formulation tools need to bring in a range of concepts and relationships to the attention of users, while on the contrary, at the construction stage, they need to keep users focused and raise only concepts and relationships that are relevant for the query task.

The challenge for query construction is to guide users to their targets with a minimum amount of deviations and backtracks. Large domain knowledge, together with top-down and bottom-up propagation of property restrictions, increases the number of possibilities enormously. At any step of query construction, users are confronted with a high number of concepts and properties to choose from. This reduces the ability of users to quickly decide on the next action.

As a solution, adaptivity and customisation are of substantial help in pruning the irrelevant concepts, paths, and properties by taking contextual factors into account (cf. [88,119]). It goes without saying that, every user is unique, yet common traits weave through them. These differences and commonalities could be addressed in a way to fit data access systems to the context of each user and user group. For instance, in an oil company, there could be different groups of users such as chemists and geologists; therefore, depending on the role of the user in an organisation, a user could only be interested in one part of the domain. An example would be the analysis of query logs on individual and group basis to derive points of interest in the conceptual space for each user and user group and to present the most relevant concepts, attributes and relationships at every stage. The approach could be improved through *recommendations* (cf. [104]), that is rather than forcing users towards one system defined direction, users could be guided through suggestions, while still being able to reach out other possibilities.

A notable example is called *reachability* in the context of this article. The navigation of conceptual space might turn into a tedious process, particularly when the *source concept* and the *target concept(s)* of an intended path are considerably distant from each other, leading to an exponential increase in the number of choices. Barzdins et al. [10,11] and Popov et al. [137] address this matter by suggesting possible paths for given source and target concepts, i.e., shortest path, which is called *non-local navigation* in this article. A better approach would be taking previous query logs into account and ranking paths accordingly, since the shortest path approach does not guarantee relevance. Soylu et al. [159] propose an approach, which exploits a query history to rank and suggest ontology elements with respect to an incomplete query that a user has constructed so far.

Various heuristic approaches should be considered to prevent any confusion that users might encounter during the navigation of a conceptual space. Ontology-based navigation might not be always intuitive for the end users, if it is not tailored adequately. For example, Soylu et al. [162], in their work regarding ontology-based Web navigation, show that navigational chains could be shortened with the help of a set of heuristics derived from the available data, such as skipping the range concept if there is only one instance available for the selected relationship (i.e., *pseudo cardinality*). From a generic perspective, observing what concepts and properties really occur at the data level is of help to eliminate the irrelevant elements at the presentation level. One could also use *sanctions* at the ontology level as suggested by Bechhofer and Horrocks [13], which are annotations that are semantically separate from the ontology and indicate for which concepts showing certain relationships or subconcepts is reasonable, valid and natural.

Combination of various search approaches could be leveraged to aid users in every phase of construction process. For instance, *keyword search* and *tag clouds* (e.g., [110,105]) could be employed to inform users about concepts, relationships, and attributes, which are not immediately available, and about possible attribute values respectively. Auto-complete facilities could serve for the same purpose. A similar approach comes from *faceted search* (cf. [175]), where users are provided with possible numbers of results at every step of the construction. Such approaches enable users to have a sense of what to expect and what direction to take, and ultimately prevent the construction of queries that are not sound – e.g., queries with no results or with a massive number of results. Regarding the active construction phase, the *redundancy* of affordances (e.g., different means to navigate to next concept) has the potential to improve user experiences and the accessibility of interface functionality by providing alternative options. *Provenance* is also worth to mention in this context; if a user loses the sense of state and track of navigation, this could have fatal implications in the user experience, which necessitate such information to be continuously available.

Collaborative query formulation, or *collaborative search* in a broader context (cf. [119,188]), and *query reuse* (cf. [177,97]) are among other possible directions. The collaboration could take place among different types of users; however, in the present context, *domain expert - IT expert* and *domain expert - domain expert* type of collaborations are of particular interest. This is because queries above a certain level of complexity could be realised either exclusively by IT-skilled users or with the help of IT-skilled users. The collaboration between domain experts may make sense in situations where the transfer of knowledge between peers is of importance. Still, one could anticipate that the medium and way of collaboration for these two types to be distinct, for instance, in domain expert - IT expert case, a domain expert is likely to work on visual level, while a database expert is likely to work at textual level, which he/she is potentially more comfortable and efficient with. Such situations require interface to be able to manage both perspectives simultaneously.

Query reuse is another form of knowledge transfer, i.e., *passive collaboration* (cf. [119]), which allows users to modify previous queries to fit them to the current task. Query reuse saves user effort and has a didactic role in the training of end users. Having said that, however, query reuse should be carefully utilised as it might lead to poorer quality query results and greater overconfidence in the correctness of results (cf. [3]). From another perspective, existing queries could be abstracted as concepts for the use of end users; such an approach not only has potential to simplify the realisation of complex tasks, but is also promising in providing extensions to the ontology on the fly (cf. [69]).

Finally, given the increasing size and velocity of data, *attention* turns out to be a precious and limited resource (cf. [132]), therefore a visual query formulation tool needs to address proactive and reactive scenarios, where data is automatically *detected*, *assessed*, and *acted upon* (cf. [69]). The challenge is to drive user attention within the large streams of data, so that valuable data fragments could be exploited. This also applies for non-stream scenarios, where new data comes in at arbitrary times. In such cases, a *push-based* approach is preferable over a *pull-based* approach (i.e., on-demand) for the users due to low cognitive load and high time-efficiency. The query construction is similar to that of a normal case, though the execution and result handling differ, since the matching data fragments and predictions based on them (i.e., proactive) are expected to cause automated actions at data or application level (i.e., reactive) (cf. [24]). Hence, query interfaces should provide intuitive means to associate queries with possible actions. This fact, along with the variety in data (e.g., temporal and spatial data), necessitates domain-specific interaction elements and modalities as well, such as range sliders and switches with haptic control, to trigger innate user reactions.

5.2.3 Discussion

A foundational issue in the development of VQSs and VQLs is to drive the capabilities of the output medium and human visual system at an optimum level, while bridging the gap between the application and the user mental model (cf. [116,140]). Primary points of attention are over perceptual, cognitive, and interactional aspects of the presentation. As evidenced by the literature, a singular approach is not sufficient to meet the diverse requirements and needs (cf. [36,93]). For this reason, *multi-paradigm* and *multi-perspective* approaches are promising to address diverse type of users, tasks, and contexts. In the former, multiple representation and interaction paradigms, each associated with the task(s) and knowledge that it is best suited to, are combined, and in the latter different query formulation approaches, such as keyword search, natural language, and formal textual query editing, are combined with visual query formulation. However, a multi-paradigm and multi-perspective approach is not sufficient alone. Although the adaptivity approach, which has an established research legacy, closes the gap, it should be used with care. The excessive use of automation could cause users to lose the sense of control and awareness and lead to frustration – i.e., *perceived user control* and *situation awareness* (cf. [157,165]).

A query formulation interface should not be considered in isolation; it should be well integrated into the organisational context with respect to the *data flow* and *life cycle*. A notable example is *instance-level navigation* (cf. [147,137]), with which users are able to navigate data, i.e., the result set of an executed query, by following links between individual instances. The particular use of instance-level navigation is *data curing* and *basic analysis*; however, it is also part of *iterative* and *exploratory search* (cf. [177,180]). In many cases, a query needs to be reformulated several times (i.e., iterative), where all iterations before the final query have an exploratory character. That is after inspecting the result set (i.e., interacting with data), a user gains more insights about the domain and data and revises his/her query accordingly. Instance-level navigation could also happen during query formulation by providing *cues* (cf. [147]), which are example results from the partial query.

Concerning the communication of extracted data, two primary notions, namely *data with context* and *data in context*, are introduced in this article. The former emphasises the need for meta information associated with the data, called *data provenance*, i.e., where it comes from (e.g., data federation), why (e.g., reasoning), its history. (cf. [89]). It is also an essential practice for dealing with *ontology evolution* (cf. [103]) to identify the source of data with respect to the changes among ontology versions. Regarding the latter, it concerns data delivery and interaction in a natural form, for instance, on a map for geospatial data. These lead us to the fact that a query formulation tool is indeed a member of a large tool portfolio, which includes other tools addressing a variety of purposes such as analysis, sense-making and collaboration. Interaction and data flow among these applications are as essential as collaboration between human actors for increasing the value creation potential. These applications could share a common framework that ensures *data interoperability* and *application interoperability*. Facilities that allow extraction of data in different formats may be provided to reach an integrated work environment as well (cf. [163,179]).

Lastly, for an *ontology-driven user interface* (cf. [14]), the challenge is not only the usability of the interface, but also the *usability of the ontology*, which steers the interface. Yet, the usability aspects of ontologies remain unnoticed to a large extent; the mismatch/gap between a user's understanding of domain and an ontology could easily hamper the success of a well-designed interface. Examples are available from a small number of studies – e.g., users expect a relationship to be an attribute or get lost while seeking a concept (e.g., [137,162]).

In an OBDA scenario, particularly in large industrial settings, ontologies are possibly to be obtained with an *ontology bootstrapping* process (cf. [164,148]), where existing documentation and database schemas are automatically harvested into ontologies. The result of the bootstrapping process is not always expected to be of high quality, since the underlying resources do not necessarily capture the rich semantics of the domain and automatic approaches have their limits. A considerable manual effort is required to enrich and fine-tune a bootstrapped ontology. Usability metrics for ontologies should also be developed to aid ontology design and presentation, for example, the ratio of the length of an expected path to the length of an observed path that a user takes to reach from a source concept to a target concept while navigating (cf. [162]).

6 Research landscape and directions

Visual query formulation is a viable alternative to complex formal textual languages and is preferable over *keyword* (cf. [23]) and *natural language interfaces* (NLI) (cf. [52,115]), which do suffer from the incapability of expressing complex information needs and natural language ambiguities respectively, for querying structured data. The research on ontology-based visual query formulation is a multi-front endeavour situated on a set of interrelated research fields, which is highlighted throughout this article and overviewed in Figure 6.

Indeed, research on visual query formulation is longstanding; however, early approaches such as *QBE* (Query by Example) [189], *QBD** (Query by Diagram) [4], *Visionary* [15], *TableTalk* [61], *HVQS* [41], *VKQS* [153], *VISUAL* [9], *X-VIQU* [43], *Xing* [62] and *XQBE* [26] employ database schemas, object-oriented models, semistructured models (e.g., XML), proprietary graph-based models, etc. Experimental research shows that approaches that rely on logical models (e.g., database schemas) are not as effective as conceptual approaches, where interaction is in terms of real world concepts [154].

An ontology-based approach has a lot to offer in this respect. However, its applications to visual query formulation are comparatively recent (e.g., [10,38,106, 11]) and existing ontology-based data access approaches are rarely supported with visual query formulation tools (e.g., [70]). In what follows, notable ontology-based or ontology-suited approaches (e.g., generic approaches for graph-based data structures) are presented and discussed first; the goal is not to provide an extensive review, but to provide a meta perspective. Then, with respect to the current research trends, a set of directions is suggested.

6.1 Research context

Existing ontology-based visual query formulation approaches could be considered in three lines of research. The first line of research is purely language-oriented (i.e., VQLs – e.g., [63,80,155,84,76,1]). The second line of research is system-oriented and utilises formal visual languages (i.e., VQSs with a VQL – e.g., [38,10]). The third line of research is purely system-oriented (i.e., VQSs without a VQL) and existing approaches largely focus on *semantic data browsers and search interfaces* for the Web (e.g., [79,27]).

The first line (i.e., VQLs), such as *Nitelight* [155] and *QueryVOWL* [76], is out of interest in the context of this article due to their high-level of formality. They are not adequate for end users as they demand high level knowledge and skills to understand syntax and semantics

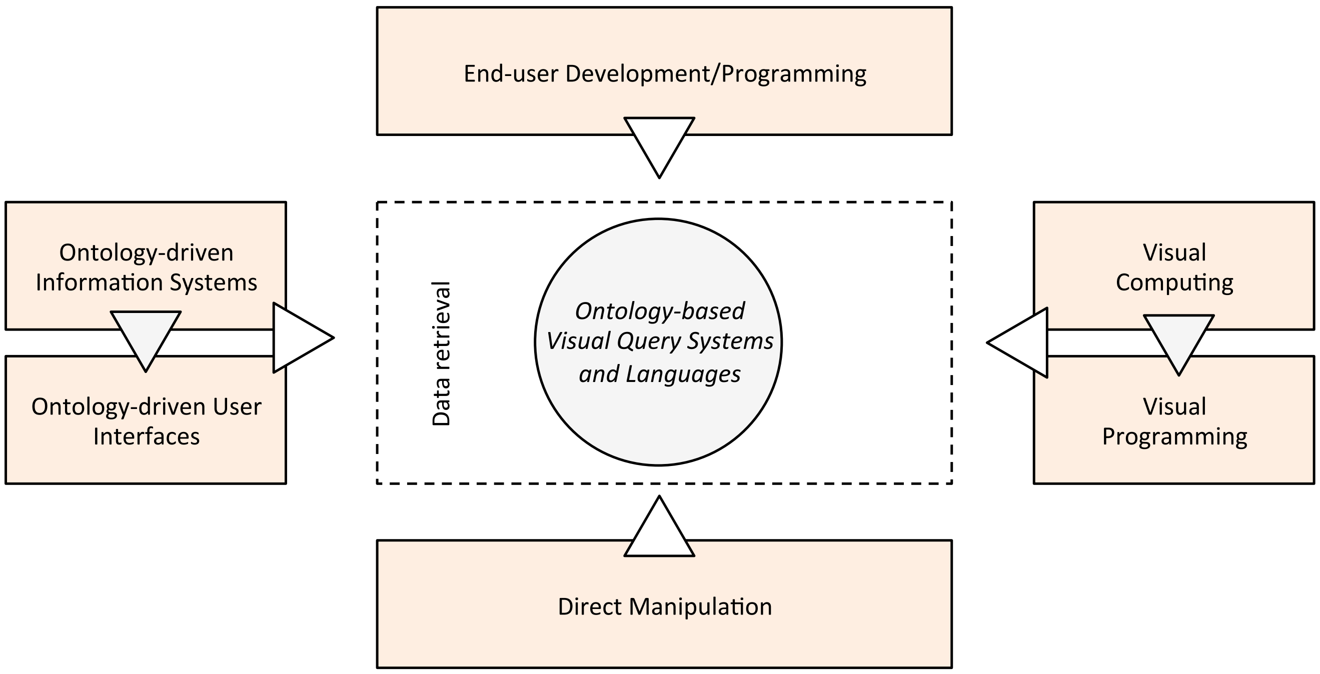

End-user Development/Programming
Ontology-driven Information Systems
Ontology-driven User Interfaces
Data retrieval
Ontology-based Visual Query Systems and Languages
Visual Computing
Visual Programming
Direct Manipulation

Fig. 6 Ontology-based visual query formulation as a multi-front endeavour.

behind a given visual notation – an example is depicted in Figure 7 for QueryVOWL. Therefore, in the rest of the article, the focus is mainly on the second and third lines of research. Semantic data browsers and search interfaces are considered in terms of VQSs, since they indeed do generate queries without disclosing them to users. The focus is particularly on the notable approaches meant for the Web although they are meant for browsing rather than querying, since they are widely used and better embraced, query and information retrieval interfaces have considerable commonalities, and the bulk body of research belongs to the Semantic Web community.

Faceted search (cf. [175]) and *query by navigation* (QbN) (cf. [171]) are two salient approaches in the Web in terms of their suitability for ontology-based query formulation and their inherent compatibility. An advanced combination of form-based and menu-based representation styles and largely range selection interaction style characterise the faceted search, with which projection and selection operations could easily be applied over a concept. The determining characteristic of QbN is the navigational interaction style, which offers an intuitive way for joining concepts through pursuing the links in between. There is a natural fit between these two approaches, as there is a fair share of primary query operations (i.e., select, join, and project). In the following, the harmony between faceted search and QbN is discussed with respect to ontologies and ontology-based examples.

6.1.1 Faceted search

Faceted search is based on a series of orthogonal dimensions that can be applied in combination to filter the information space. Each dimension, called *facet*, corresponds to a taxonomy, that is each instance of a concept is classified with respect to multiple explicit dimensions (i.e., *faceted classification*). A taxonomy, i.e., facet, could have a hierarchical structure, named *hierarchal facets*. Constraints associated with different facets are mostly combined with logical "AND" (cf. [175]). For example, for a concept that describes "cars", facets could be the "brand", "type", "year", "mileage", etc. The ideas that drive faceted search likely emanate from the work on *dynamic queries* [151] and *view-based search* [136], and inspire the work on *instantaneous response interfaces* [127].

The work on dynamic queries is based on the interactive control of visual query parameters leading to a rapid display of search results. The idea is further improved by taking the relationship between data distribution and user selections into account in a proactive way in order to limit the user to a set of satisfiable parameter combinations (cf. [2, 56]). For instance, given a movie database, upon the selection of a particular actor, all the dates, for which the selected actor has no movie, are excluded from available options and/or the number of available instances is displayed next to each possible date option. The number of achievable instances upon a possible selection of a facet or facet option is sometimes

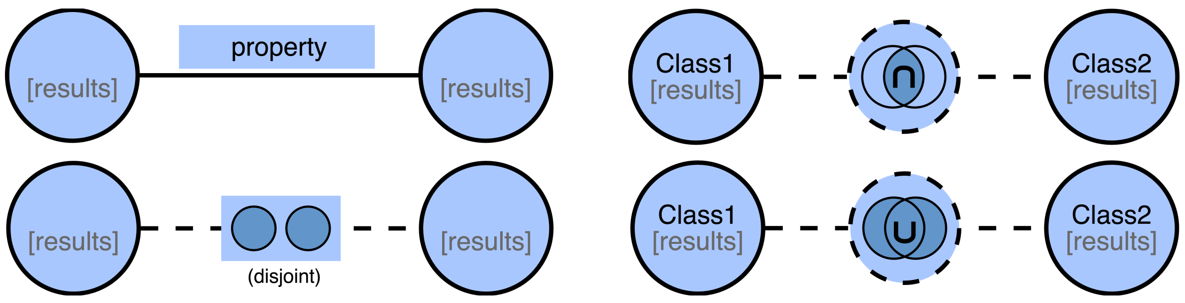


Fig. 7 An example fragment of QueryVOWL – a visual query language for linked data.

called *numeric volume indicators* (NVIs) and is an integral part of faceted search [181]. In view-based search, a view corresponds to a facet and the difference from the earlier work is the use of hierarchical facets, while the work on instantaneous response interfaces relies on the idea of providing instant suggestions at schema level and data level to the user as he/she interacts. Research-wise, *Flamenco* [184], *RB++* (Relation Browser) [187], *mSpace* [147], and *Exhibit* [87] are well-known examples.

Flamenco[3] (see Figure 8) combines keyword and faceted search. Initially all facets are shown and, after selecting a facet or executing a keyword search, the results are presented and facets are displayed in a column next to the result area. If the selected facet has subcategories, they are shown and if it has a hierarchical structure, the subcategories of the selected option are shown. RB++ lists all facets and their entire values at the top part of the interface and the result set is displayed at the bottom part. Facets can be reordered; however, this has no effect on filtering.

mSpace provides keyword search support and presents four panels. The first panel, *facet browser*, displays active facets in a linear form. The order of facets matters, since the filtering is done left to right. A user can make multiple selections within a facet, which are combined with logical "OR". The second panel, *interests*, serves as a bookmark facility, with which the user can retain instances of interest. The third panel, *information*, presents information about the last selection, i.e., instance or facet. The last panel, *preview cue*, provides example instances during the search.

Exhibit is similar to commercial faceted search interfaces; it consist of two panels: *browse panel* and *view panel*. The browse panel presents a set of facets, while the view panel presents the search results. The view panel is configurable, that is different view options are possible such as lists and maps. All faceted search interfaces described here are accompanied with NVIs and the trail of selections (aka *breadcrumbs*) for user awareness and control.

6.1.2 Query by navigation

QbN exploits the graph-based organisation of data to allow the user to construct queries by traversing the relationships between concepts. The type of a relationship could be a simple *is-a* relationship (i.e., concept-subconcept) or more general. An early attempt for the application of QbN comes from Ter Hofstede et al. [171]; their work combines *stratified hypermedia* (cf. [30]) and QbN, which are particularly well-known in document retrieval domain, and applies it to traditional database systems by exploiting the parallels in *information disclosure* between document retrieval systems and traditional database systems.

Stratified hypermedia is an architecture in which information is organised via several layers of abstraction. The base layer contains the actual data (i.e., *hyperbase*), while other layers contain the abstraction of this data (i.e., *hyperindex*) and enable access to the base layer. In a document retrieval system, the abstraction layer is composed of hierarchically organised keywords; an *indexing* process is required to construct the abstraction layer and to *characterise* the documents (i.e., identification of documents with respect to the abstraction layer). The characterisation process is known to be a hard problem; however, in traditional database systems, the characterisation of data is directly given by a reference model [171]. Ter Hofstede et al. [171], in their work, built their abstraction layer on the top of *Object Role Modelling* (ORM) (cf. [77]), due to its closeness to the real world compared to the *Entity-Relationship* (ER) models (cf. [59]), and employ QbN to enable the user to navigate hyperindex and construct linear paths. Although the resulting branches, with pure QbN, are connected with logical "AND", the authors provide a structure editor, with which the user could combine linear paths in advanced ways including negation and logical "OR".

[3] http://flamenco.berkeley.edu

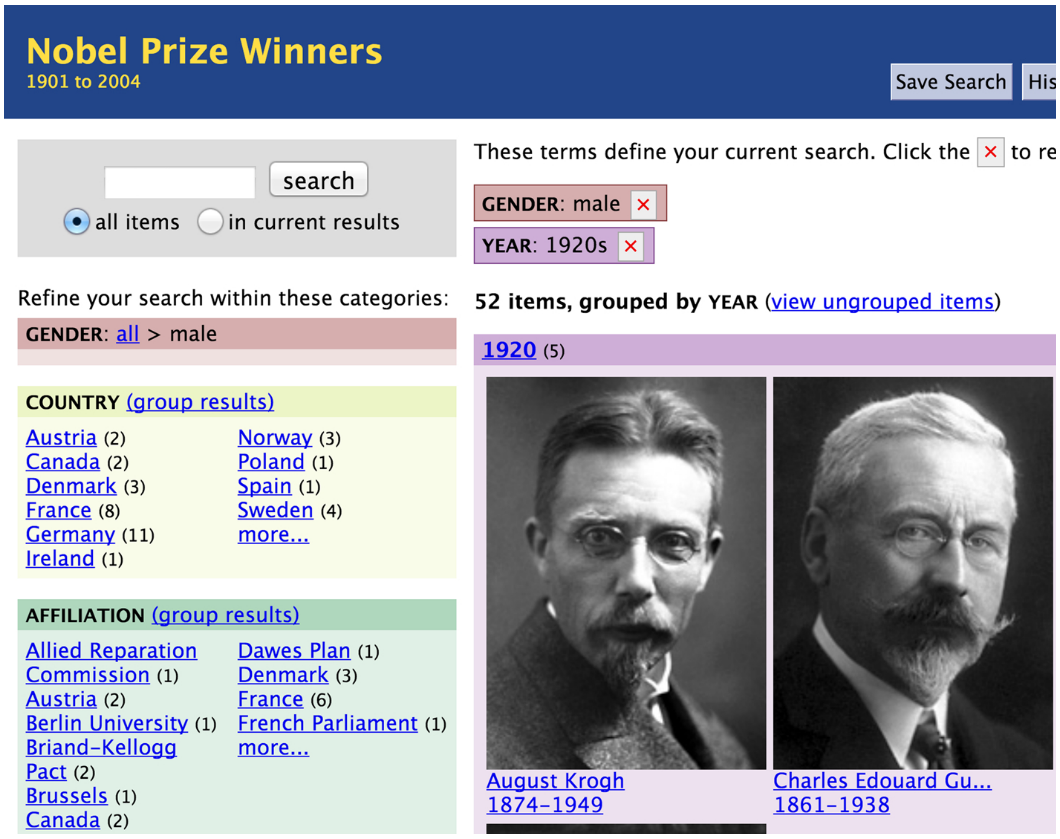


Fig. 8 Flamenco – a faceted search interface.

Concerning the present research, QbN is typically realised in two forms, which are characterised with menu-based and diagram-based representation styles respectively. Examples of QbN with menu-based representation style are usually the semantic data browsers developed for querying and browsing linked data (cf. [22]). Particular examples are *Tabulator* [16], *TcruziKB* [122], and *SWC* [162]. Tabulator provides an instance level navigation experience and employs a tree-based approach (i.e., indented lists), where a user can expand and retract nodes and follow relationships by selecting tree elements. Unlike Tabulator, TcruziKB does not display the whole hierarchy and the goal is query construction; a user starts with a keyword search to find a *kernel concept* (i.e., starting concept) and, once found, system suggests relationships. The selection of a relationship is followed by a selection of a range concept, then the system moves focus to the selected concept; this process is continued, until the query of interest is complete. The operation of moving from one concept to another (i.e., changing focus) is known as *pivoting operation* and the active/focus concept is called *pivot* (cf. [137, 100]).

SWC[4] is for instance level mobile navigation (see Figure 9), and nitially provides a flat list of concepts; upon selection of a concept, the tool presents the list of subconcepts or list of available instances. Once an instance is selected, available relationships are shown, with which the user can change focus to another instance.

Examples of QbN with diagram-based representation style are *TAMBIS* [167], *SEWASIE* [38], *GQL* (Graphical Query Language) and *ViziQuer* [10, 11, 190], and *Visor* [137]. TAMBIS has a diagram-based pane, where a user query is represented as a tree. The user starts with a kernel concept, as a root node, and continues construction by expanding nodes through their properties. The user can type in his constraints for an attribute directly on the corresponding leaf node. The tool developed within the context of SEWASIE project has a query manipulation pane, which provides a graph-based representation of the constructed query. The query manipulation pane maintains the focus and suggests the user possible concepts and relationships with drop-down menus.

GQL is a graphical query language built on SPARQL and OWL, while ViziQuery is a query system driven

[4] http://www.ahmetsoylu.com/pubshare/icae2011/

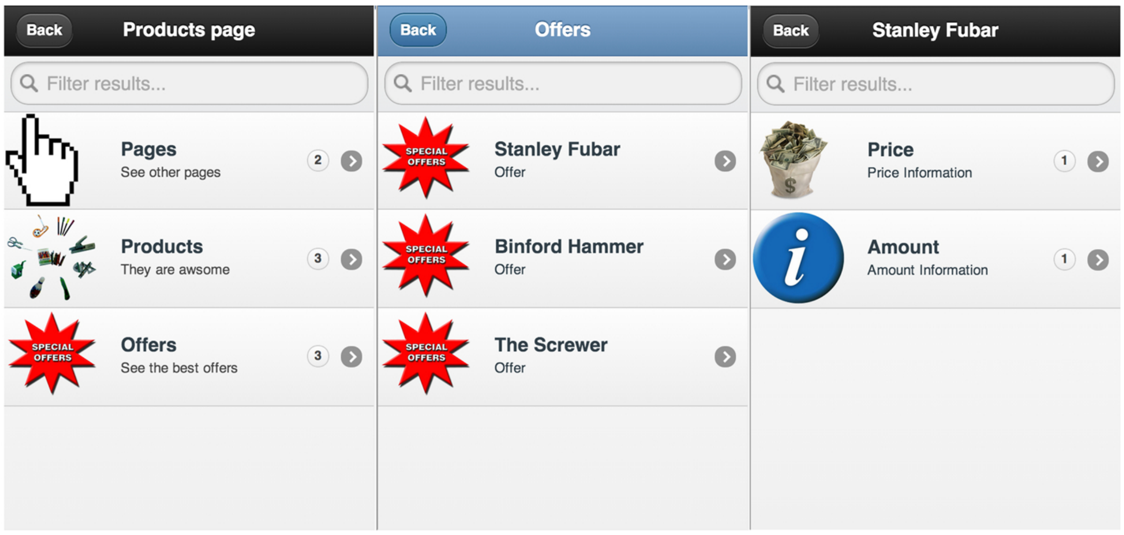


Fig. 9 SWC – an instance-based mobile data browser.

by GQL. Similar to SEWASIE project, ViziQuer also offers a graph-based pane for query manipulation; the tool firstly presents a list of concepts, so that the user can select a kernel concept. Upon the selection of a kernel concept, the user can add a subordinate concept from the list. The system offers the shortest or the most likely paths between the kernel concept and the subordinate concept, or the user can draw links between any concepts to receive path recommendations. It is possible to navigate from concept to concept by directly following relationships as well, though the mechanism is not clearly exemplified or explained. For the selection and projection operations, ViziQuer provides a form-based dialog.

Visor initially provides a list of all concepts (flat or hierarchical views are available) and a diagram-based query pane. User-selected concepts are moved to the query manipulation pane. The user can select as many concepts as needed (called *multi-pivot* approach). The tool automatically links all the concepts. A blue circle, with a number in it, is displayed in the middle of each edge; this number represents the number of relationships that link two concepts. If there are no relationships that link two concepts, the first shortest path is found and intermediary nodes are added. These facilities are used for ontology exploration and to derive a subgraph of interest, while full query construction is handled through a form and menu-based approach with respect to the derived subgraph of the ontology.

6.1.3 Hybrid approaches

Faceted search is widely used in commercial websites, such as eBay and Amazon, for listing and filtering products and menu-based navigation is the backbone of Web browsing; therefore, a typical Web user is expected to be familiar with faceted search and QbN. Besides, faceted search and QbN have a natural harmony with ontologies, since the representation of taxonomies and associations between concepts are in the core of ontologies. However, considering each approach in isolation, faceted search, in its most common form, breaks down as soon as a join between several concepts is required. Ontologies are richer and represent more complex relationships between concepts than child-parent relationships. Similarly, QbN is good at exploring such links between domain concepts, yet it needs facilities to accommodate the selection and projection operations. In this respect, the amalgamation of faceted search and QbN is a very sensible direction.

Indeed, faceted search includes a navigational flavour; this is mostly due to fact that it allows the user to drill down within hierarchical concept and facet taxonomies. For this reason, it is sometimes called *faceted navigation* or *faceted browsing*. Tunkelang and Marchionini [175] distinguish faceted search and faceted navigation to highlight the difference between the navigational and range-based aspects of faceted search. However, such a distinction is inherently redundant, since every selection within a facet results in a search and the navigational aspects of faceted search are not truly dominant.

Structurally, query construction on a hybrid of QbN and faceted search corresponds to a navigation and pruning process within a *network of hierarchies*; an example is given in Figure 10. There are two concepts in this example, namely “Student” and “Course”, which are associated through “takes” relationship. Each concept has a set of facets, for instance “Type”, “Title”, “Year”, and “Status” for the “Course”. These facets are derived from attributes, and they could also be derived from relationships as shown in Figure 10 for the "takes" relationship acting as a facet for "Student" over the representative "Title" attribute of "Course" concept. The same relationship is also used for navigating from "Student" to "Course" concept or vice versa. One should realise that deriving facets from a concept is not always straightforward, one may require some processing, for instance for the discretisation of numeric attributes (e.g., GPA). Hybrid approaches in the current literature could be considered in twofold, which are built on menu-based and diagram-based representation styles as well.

Menu-based hybrid approaches are mostly built by introducing pivoting into faceted search interfaces, so that the user can navigate between related data sets. Examples of menu-based hybrid approaches are *Parallax* [86], *Humboldt* [100], *Rhizomer* [27], *VisiNav* [79], and *tFacet* [28]. Parallax and Humboldt are generic approaches for graph-based data structures with applications on Web data. In Parallax, the user starts with a keyword search, after which a set of matching concepts and instances are offered to the user. Upon selection of a particular concept, all available instances are displayed along with a set of facets. The interface also displays a set of relationships, with which the user can browse from one set of instances to another.

In Humboldt, as soon as a data source is loaded, all available instances are presented as a list, and types of these resources are listed in a tag cloud to filter the resources. Differently, in Humboldt, facets are based on relationships and grouped by type. There could be various relationships between two concepts; however, rather than deriving a facet for each relationship, only a single facet is derived per associated concept. Such an approach results in a decrease in the number facets, while introducing ambiguity. The approach also provides a mechanism for navigating between concepts, which is supported with an animation to prevent user confusion.

Rhizomer[5] (see Figure 11) is a tool for exploring semantic Web data and is similar to Parallax; it follows overview, zoom and filter mantra and uses techniques, such as navigation menus, tree-maps or sitemaps, to provide an overview of the dataset. VisiNav is built on four operations, namely *keyword search*, *object focus*, *path traversal*, and *facet specification*. The keyword search provides an initial set of results and with object focus the user can view a single result. The path traversal allows the user to reach out other concepts either through a single instance or concept, while with facet specification the user can filter down the result set.

tFacet[6] is a tool for hierarchical faceted exploration of RDF data. The user first needs to select a base concept, for which the exploration is to be limited to, then the interface, as shown in Figure 12, displays facets at the left-hand side in a tree form (i.e., indented list). The first level of the tree represents the facets derived from the direct relationships and attributes of the base concept, while the children of a tree item represent the facets of another concept associated through the relationships that the parent tree item represents (e.g., in Figure 10, "Student" being the base concept, "takes" property would be a direct facet, while "Year" attribute would be a child of it). Selected facets appear in the bottom middle part of interface along with possible options. Options selected within a facet are connected with the "OR" connective, while options selected from different facets are connected with the "AND" connective. The result set is continuously displayed at the top part of interface; the user can add/remove attributes for the result set by using a dialog activated with "visible properties" button. Although a tree-based approach can represent a large number of direct and indirect facets for a concept, for large ontologies the depth of the tree is an issue. This is because each tree item is expected to have a high number of children hindering the usability (cf. [93]).

An important problem with menu-based hybrid approaches is their inability to aggregate information from different concepts (i.e., result set is formed by a single concept). Another problem is poor support for overview (i.e., a global view of connected concepts, constraints imposed, and attributes selected for the output).

Concerning the diagrammatic approaches, notable examples are *OZONE* [170], *MDDQL* [92], *gFacet* [82] and OptiqueVQS [160]. In OZONE, the user first searches for a kernel concept from a hierarchical list of concepts. Once a concept is selected, it appears in the diagram-based query manipulation pane in the form of a rectangle. Each node is the aggregation of a concept name and its properties. The navigation from one concept to another happens through the expansion of relationships. OZONE mainly uses a form-based representation style, that is a user has to type in constraints for attributes. However, the user can specialise and generalise concepts in a fashion similar to the faceted search, by clicking on a node and selecting a more general or more specific concept type.

MDDQL comes with four panes: a diagram-based pane for representing queries in tree form, a suggestion pane listing possible terms (e.g., concepts and properties), and two other panes for attributes that are constrained and selected for the output respectively. A

[5] http://rhizomik.net/html/rhizomer/

[6] http://www.visualdataweb.org/tfacet.php

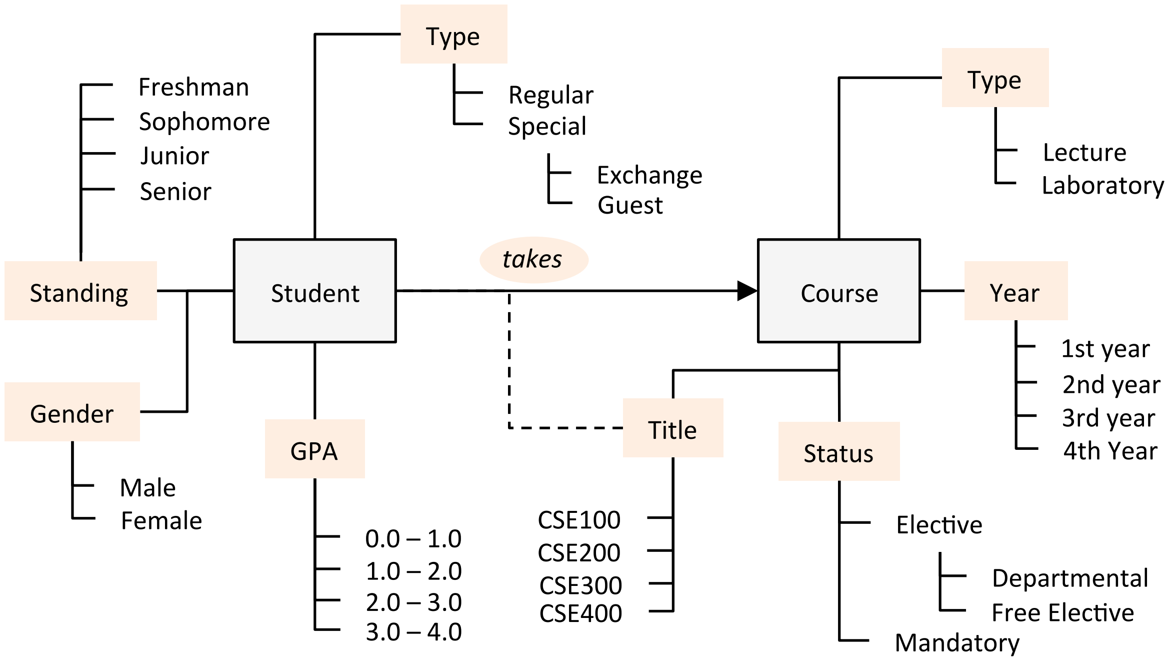
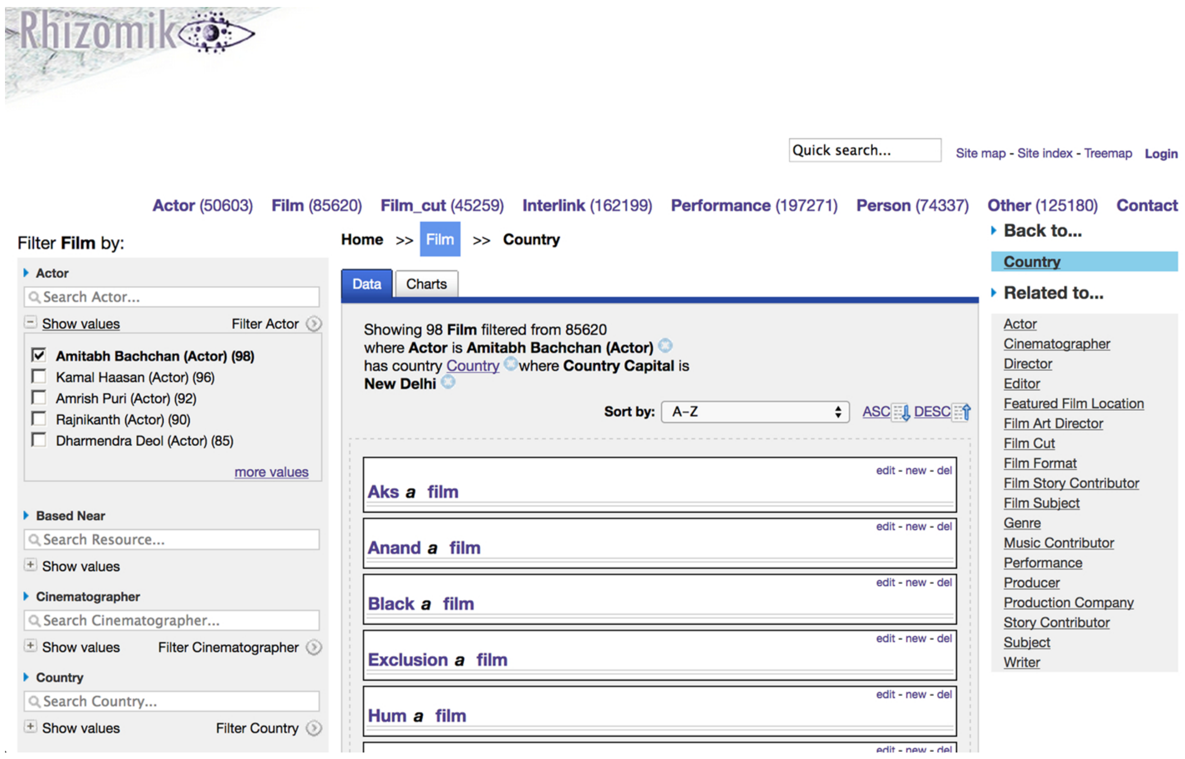
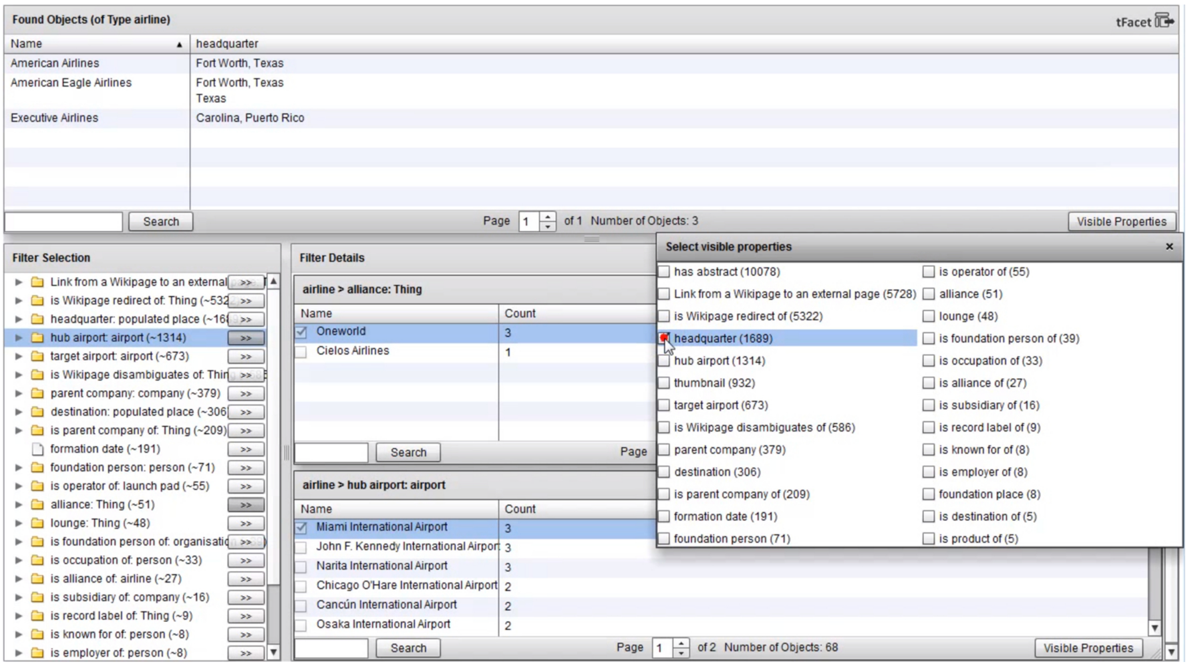


Fig. 10 QbN and faceted search – a network of hierarchies.



Showing 98 **Film** filtered from 85620
where **Actor** is **Amitabh Bachchan (Actor)**
has country Country where **Country Capital** is
New Delhi

Fig. 11 Rhizomer – a tool for exploring semantic data.



Found Objects (of Type airline)

Name ▲	headquarter
American Airlines	Fort Worth, Texas
American Eagle Airlines	Fort Worth, Texas Texas
Executive Airlines	Carolina, Puerto Rico

Page 1 of 1 Number of Objects: 3

Filter Selection

- Link from a Wikipage to an external
- is Wikipage redirect of: Thing (~532
- headquarter: populated place (~16
- hub airport: airport (~1314)
- target airport: airport (~673)
- is Wikipage disambiguates of: Thin
- parent company: company (~379)
- destination: populated place (~306
- is parent company of: Thing (~209)
- formation date (~191)
- foundation person: person (~71)
- is operator of: launch pad (~55)
- alliance: Thing (~51)
- lounge: Thing (~48)
- is foundation person of: organisati
- is occupation of: person (~33)
- is alliance of: airline (~27)
- is subsidiary of: company (~16)
- is record label of: Thing (~9)
- is known for of: person (~8)
- is employer of: person (~8)

airline > alliance: Thing

Name	Count
✓ Oneworld	3
Cielos Airlines	1

airline > hub airport: airport

Name	Count
✓ Miami International Airport	3
John F. Kennedy International Airpor	3
Narita International Airport	3
Chicago O'Hare International Airport	2
Cancún International Airport	2
Osaka International Airport	2

Page 1 of 2 Number of Objects: 68

Select visible properties

☐ has abstract (10078)	☐ is operator of (55)
☐ Link from a Wikipage to an external page (5728)	☐ alliance (51)
☐ is Wikipage redirect of (5322)	☐ lounge (48)
✓ headquarter (1689)	☐ is foundation person of (39)
☐ hub airport (1314)	☐ is occupation of (33)
☐ thumbnail (932)	☐ is alliance of (27)
☐ target airport (673)	☐ is subsidiary of (16)
☐ is Wikipage disambiguates of (586)	☐ is record label of (9)
☐ parent company (379)	☐ is known for of (8)
☐ destination (306)	☐ is employer of (8)
☐ is parent company of (209)	☐ foundation place (8)
☐ formation date (191)	☐ is destination of (5)
☐ foundation person (71)	☐ is product of (5)

Fig. 12 tFacet – a tool for hierarchical faceted exploration of RDF data.

user can start with a kernel concept, as the root node of a query tree, and keep expanding nodes by selecting a property from the suggestion pane.

gFacet[7] (see Figure 13) is a tool for exploring semantic data and is similar to OZONE. The search process starts with a keyword search for finding a kernel concept; once a kernel concept is selected, it appears on a diagram-based query manipulation pane. The facets in gFacet are only the ones derived from relationships between concepts (e.g., "takes" for "Student" in Figure 10); the user can keep linking concepts and filtering the result set by expanding relationships into new nodes and selecting values for them. The result set attributes only come from the kernel concept and the concepts added to query graph by navigation are merely used for filtering the result set. However, the user has the opportunity to change the kernel concept.

OptiqueVQS (see Figure 14) relies on a widget-based architecture so as to exploit multiple coordinated representation and interaction paradigms for graph navigation and facet refinement. A graph-based widget is meant to provide an overview, while a menu-based widget and a form-based widget deal with navigation and facet manipulation and provide view/focus. The diagrammatic query representation is informal, free off SPARQL jargon, and simplified (i.e., unidirectional and tree-shaped) – cf. Figure 14 vs. Figure 7.

[7] http://www.visualdataweb.org/gfacet.php

The problem of inability to aggregate information from different concepts generally persists for diagram-based hybrid approaches, while a better overview is provided, since diagram-based approaches are naturally good at providing a global view. However, this time the problem is usually poor support for focus, that is ability to channel user to a specific part of an active task.

6.2 Discussion and directions

A good deal of existing work, with the rise of the Semantic Web technologies, targets semantic data browsing and searching on the Web. Although early approaches, such as Tabulator, are mostly suitable for IT skilled users, successor approaches address end users and provide valuable techniques and insights for the use of ontologies for enhanced end user experiences in data access.

However, firstly, these approaches are mostly very minimalistic in expressivity both in terms of queries and domain knowledge they can express. In many cases, even conjunctive queries are not well established and available domain knowledge is very poor, since it is extracted from instances (i.e., instance-oriented) rather than an ontology. In a traditional scenario, as discussed earlier, certain types of domain knowledge and queries should be addressed or IT expert support should be possible to enable end users to closely describe and find data of interest within Big Data sources.

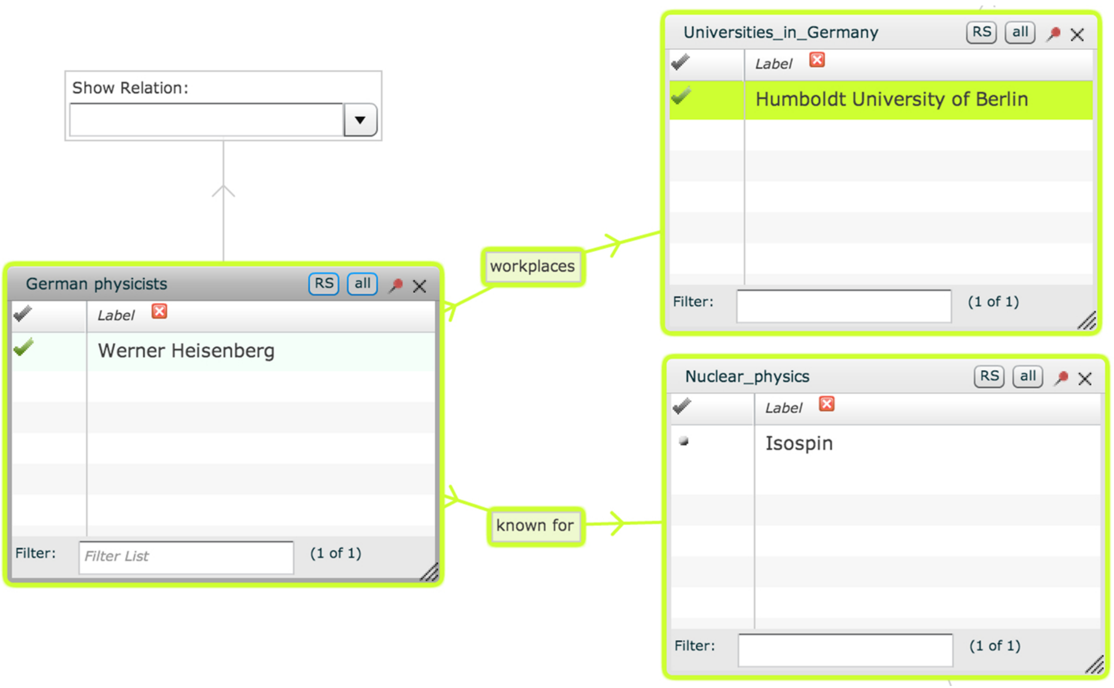


Fig. 13 gFacet – a tool for exploring semantic data.

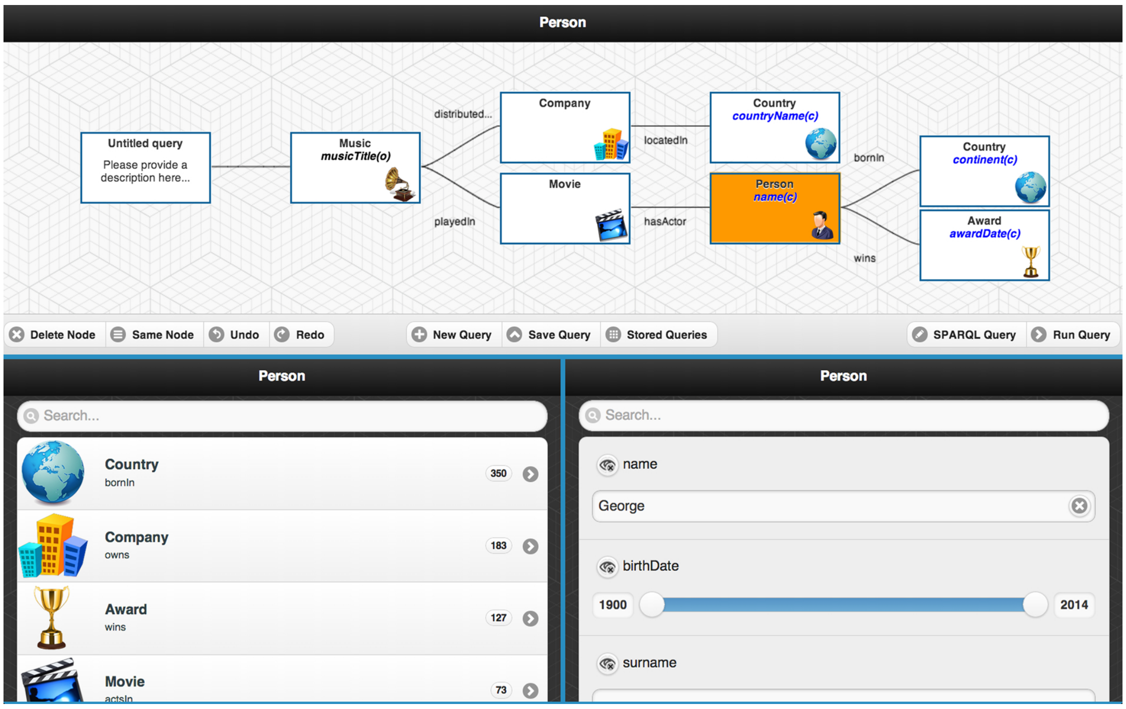


Fig. 14 OptiqueVQS – an ontology-based visual query system.

Secondly, semantic data browsing on the Web is highly exploratory and requires instant and continuous interaction with data (i.e., *database-intensive*). This is contrary to traditional data access systems, where a user first constructs the information need with the terms coming from a schema and then retrieves the associated data. As discussed earlier, interaction with the real data at every phase of query formulation is of use for understudying the organisation of underlying data and for improved user experiences. Yet, the problem with this approach in a traditional system, particularly in an industrial setting, is linked with the sheer volume and velocity of data. It is not always feasible to maintain a constant and frequent contact with voluminous and stream data sources. The same applies to NVIs for faceted search interfaces, indeed every possible facet selection and every possible join with a new concept means a new partial query, which should be run against the data sources to provide the user with the number of results for each possibility. This situation opens up yet another research challenge, for which not only exact solutions but also approximations could be of a great use.

As evidenced by the presented hybrid approaches, the combination of faceted search and QbN is a promising and natural direction. However, there are certain issues to be addressed for its applicability in a traditional query formulation setting. Presented approaches are mostly prototypical and none of them is scalable to an industrial setting. There are no means provided to tackle large ontologies. Query formulation remains very implicit and limited due to their high exploratory nature and the line between query construction and exploration is not clear. For a data browsing scenario, this blurry line may not be a problem while tackling with data in small sizes, where data is instantly provided at every phase of interaction; however, in a traditional data access setting, the user focus is on the formal description of information need, which necessitates a clear distinction between explorative and constructive actions. In contrast with data browsers, existing query formulation tools have a strong focus on construction, while leaving exploration almost non-addressed. Exploration and construction should be addressed and intertwined clearly and user evaluations should assess both perspectives, for instance, by not only using exactly defined queries (i.e., for query writing and reading tasks), but also involving vaguely described information needs (i.e., *explorative tasks*) to assess the exploration support.

Similarly, a possible adaptation for traditional data access systems would also necessitate a strong support for view and overview, that is the user should be able to keep a constant possession of the overall state with respect to task at hand and at the same time should be able to maintain a focused engagement with the system for the active phase of the task. Compared to the data browsing and querying on the Web, in industry, information needs are more sophisticated; therefore, complex analysis and processing on extracted data is required. Presented approaches mostly rely on one or two different representation paradigms both for result visualisation, exploration and construction, with one paradigm being highly dominant. Often, a *naked object* approach is employed (cf. [79]) for presenting results (i.e., without any type specific styling). However, an extensible architecture is crucial for the incorporation of alternative and complementary paradigms as well as for introducing various other tools, such as scratchpads, bookmarks, and domain-specific visualisation components, for different purposes.

A *user-interface mashup* (UI mashup) approach (cf. [163,60]) is promising for the construction of extensible and flexible query formulation tools. The mashup idea, in the present context, is grounded on the possibility of combining the functionality and data of a set of individual applications in a common graphical space and underpins the multi-paradigm and multi-perspective approaches. *Widgets*[8] are the building blocks of UI mashups, where each widget corresponds to a standalone application with less complex functionality and presentation compared to full-edged applications. In query formulation scenario, a set of widgets could be employed, for instance, one for QbN and one for faceted search for handling query construction, one for representing results in a table, and one for visualising the results in a graph.

Soylu et al. [163] propose an architecture for widget-based UI mashups. A widget environment that provides basic communication and persistence services to the widgets manages widgets. The *orchestration of widgets* relies on the requirement that each widget discloses its functionality to the environment through a client side interface and notifies other widgets in the environment and/or the widget environment upon each user action. Then, either each widget decides on what action to execute in response, by considering the *syntactic or semantic signature* of the received event, or the environment decides which widgets to invoke with which functionality. The core benefits of such an approach are: it becomes easier to deal with the complexity, since the management of functionality and data can be delegated to different widgets; each widget can employ a different visualisation paradigm that best suits its functionality; widgets can be used alone or together, in different combinations, for different contexts and experiences (e.g.,

[8] http://www.w3.org/TR/widgets/

widgets running on distributed devices – *distributed user interfaces* [67]); and, the functionality of the overall interface can be extended by introducing new widgets (e.g., for result visualisation).

As stated earlier, the usability of underlying ontology also matters. The fact that data browsing approaches for the Web rely on instances rather than a particular schema or ontology introduces an advantage for ontology-based data access systems. From an end user perspective, the availability of an ontology implies that indeed end users can interfere with the ontology in order to align it with their own understanding of the domain and keep it up-to-date with respect to their needs. One possible way could be through IT experts, who could incorporate the feedback from end users in the design of the ontology. However, direct end user involvement may be also possible, through an approach called *query-driven ontology extensions* (cf. [69]). The goal is to enable end users to propose/extend an ontology by using the facilities attached to a query formulation interface, such as adding synonyms and basic extensions; an example could be the incorporation of a query into ontology hierarchy as a new concept. However, necessary caution has to be taken in order to avoid overloading the ontology and to ensure safeness (e.g., [90,91]).

Such a user-driven perspective could be employed for improving the different aspects of a visual query formulation tool as well, which is called *user-driven evolution* in this article. For instance, it is a good practice for a VQS or VQL to supplement its vocabulary, coming from an ontology, with end user targeted labels, descriptions and icons. Obviously, this is a tedious and long process, if not endless, for large ontologies. However, in a user-driven scenario, a visual query formulation tool may initially include icons, labels, descriptions etc. for a limited set of most commonly used vocabulary elements, while enabling end users to incorporate their own supplementary content to ontology (e.g., tagging). This approach considers a visual query formulation tool and ontology as an output of work process, over years of use, rather than a mere input.

7 Conclusions

Much work has been carried out on visual query formulation; early attempts rely on low-level abstractions and logical models such as schemas and object role modelling, yet they successfully establish the fundamentals of research domain. Recent approaches, with the emergence of the Semantic Web and OBDA, employ ontologies as a natural medium of access to traditional and open data sources available on the Web. However, the ever-increasing volume, complexity, velocity and variety of data, called Big Data, render the end-user data access problem even more challenging. In this regard, the current work is an attempt to provide a broad meta-overview on ontology-based visual query formulation, challenges, directions, and related literature. In this context, the article addresses a broad audience of researchers and practitioners interested in intuitive end-user visual query formulation interfaces.

As the discussions suggest, there are two main pillars of visual query formulation, namely expressiveness and usability; however, even the expressiveness should be considered with a usability perspective. General usability challenges, concerning the common perceptual and cognitive issues, are mostly a matter of innovative design, where appropriate paradigms should be selected with a multi-paradigm perspective. The expressivity, usability and Big Data effect, at conceptual and data level, should be handled with various approaches such as adaptivity and customisation, collaboration, and intelligent support. A query formulation tool should not only scale against data and conceptual model, but also against the personal and organisational context, which necessitate a flexible and modular architecture.

It has been also been argued that the matter is not only usability of the query formulation system or language, but also the underlying ontology, for which necessary measures should be taken for addressing possible problems and needs with an active end-user involvement. Concerning the present research, the combination of faceted search and QbN is quite promising; however, current examples diverge in terms of their goals and context (i.e., data browsing at instance level on the Web). Hence, it is necessary to realise a set of adaptations, before these approaches could be used for traditional systems with success.

One should be aware that the research on end-user visual query formulation is inevitably a usability challenge. The early pioneers (e.g., [89,36]), in the database domain, report their experiences on how underestimating the contribution of users drove them into failure. Researchers, particularly coming from the Semantic Web domain, should be aware of this and not to relearn the same old lesson. A user-centred design practice, in which many tests with small groups of users are conducted in different contexts (cf. [129]), is the key in this respect.

The insights gained as a result of this article is distilled into a set of quality attributes and features [158] and utilised in the development of OptiqueVQS. It is evaluated both with casual users [160] and domain experts [161] and acquired positive results. OptiqueVQS is part of an end-to-end semantic data access platform, namely Optique, which includes components for *query formulation support*, *ontology and mapping management*, *query transformation*, *times and streams*, and *distributed query execution* [69, 96, 70].

References

1. OpenLink iSPARQL. URL `http://dbpedia.org/isparql/`
2. Ahlberg, C., Shneiderman, B.: Visual information seeking: tight coupling of dynamic query filters with starfield displays. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1994), pp. 313–317. ACM (1994). DOI 10.1145/191666.191775
3. Allen, G., Parsons, J.: Is Query Reuse Potentially Harmful? Anchoring and Adjustment in Adapting Existing Database Queries. Information Systems Research **21**(1), 56–77 (2010). DOI 10.1287/isre.1080.0189
4. Angelaccioa, M., Catarci, T., Santucci, G.: Query by diagram: A fully visual query system. Journal of Visual Languages and Computing **1**(3), 255–273 (1990). DOI 10.1016/S1045-926X(05)80009-6
5. Angles, R., Gutierrez, C.: The Expressive Power of SPARQL. In: Proceedings of the 7th International Conference on the Semantic Web (ISWC 2008), *LNCS*, vol. 5318, pp. 114–129. Springer (2008). DOI 10.1007/978-3-540-88564-1_8
6. Assmann, U., Zschaler, S.: Ontologies, Meta-models, and the Model-Driven Paradigm. In: C. Calero, F. Ruiz, M. Piattini (eds.) Ontologies for Software Engineering and Software Technology, pp. 249–273. Springer-Verlag (2006). DOI 10.1007/3-540-34518-3_9
7. Athanasis, N., Christophides, V., Kotzinos, D.: Generating on the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). In: Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), *LNCS*, vol. 3298, pp. 486–501. Springer (2004). DOI 10.1007/978-3-540-30475-3_34
8. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
9. Balkir, N.H., Ozsoyoglu, G., Ozsoyoglu, Z.M.: A Graphical Query Language: VISUAL and Its Query Processing. IEEE Transactions on Knowledge and Data Engineering **14**(5), 955–978 (2002). DOI 10.1109/TKDE.2002.1033767
10. Barzdins, G., Liepins, E., Veilande, M., Zviedris, M.: Ontology Enabled Graphical Database Query Tool for End-Users. In: Proceedings of the 8th International Baltic Conference on Databases and Information Systems (DB&IS 2008), *Frontiers in Artificial Intelligence and Applications*, vol. 187, pp. 105–116. IOS Press (2009). DOI 10.3233/978-1-58603-939-4-105
11. Barzdins, G., Rikacovs, S., Zviedris, M.: Graphical Query Language as SPARQL Frontend. In: Proceedings of the 13th East-European Conference (ADBIS 2009), pp. 93–107 (2009)
12. Batini, C., Catarci, T., Costabile, M.F., Levialdi, S.: Visual Query Systems: A Taxonomy. In: Proceedings of the IFIP TC2/WG 2.6 2nd Working Conference on Visual Database Systems II, pp. 153–168. North-Holland Publishing Co. (1992)
13. Bechhofer, S., Horrocks, I.: Driving User Interfaces from FaCT. In: Proceedings of the International Workshop on Description Logics (DL 2000), *CEUR Workshop Proceedings*, vol. 33, pp. 45–54. CEUR-WS.org (2000)
14. Bechhofer, S., Stevens, R., Ng, G., Jacoby, A., Goble, C.: Guiding the user: an ontology driven interface. In: Proceedings of the User Interfaces to Data Intensive Systems, pp. 158–161. IEEE Computer Society (1999). DOI 10.1109/UIDIS.1999.791472
15. Benzi, F., Maio, D., Rizzi, S.: VISIONARY: a Viewpoint-based Visual Language for Querying Relational Databases. Journal of Visual Languages and Computing **10**(2), 117–145 (1999). DOI 10.1006/jvlc.1998.0102
16. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI 2006) (2006)
17. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American **284**(5), 34–43 (2001)
18. Beshers, C., Feiner, S.: Auto Visual: Rule-Based Design of Interactive Multivariate Visualizations. IEEE Computer Graphics and Applications **13**(4), 41–49 (1993). DOI 10.1109/38.219450
19. Besnard, P., Cordier, M.O., Moinard, Y.: Ontology-based inference for causal explanation. Integrated Computer-Aided Engineering **15**(4), 351–367 (2008)
20. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing **6**(2), 161–180 (2010). DOI 10.1016/j.pmcj.2009.06.002
21. Bevan, N., Macleod, M.: Usability measurement in context. Behaviour and Information Technology **13**(1-2), 132–145 (1994). DOI 10.1080/01449299408914592
22. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems **5**(3), 1–22 (2009). DOI 10.4018/jswis.2009081901
23. Bobed, C., Esteban, G., Mena, E.: 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 10.3233/KES-130255
24. Boley, H., Kifer, M., Pătrânjan, P.L., Polleres, A.: Rule Interchange on the Web. In: Proceedings of the 3rd International Summer School Conference on Reasoning Web (RW 2007), *LNCS*, vol. 4636, pp. 269–309. Springer (2007). DOI 10.1007/978-3-540-74615-7_5
25. Borst, W.N.: Construction of Engineering Ontologies. Ph.D. thesis, University of Twente, Enschede (1997)
26. Braga, D., Campi, A., Ceri, S.: XQBE (XQuery By Example): A visual interface to the standard XML query language. ACM Transactions on Database Systems **30**(2), 398–443 (2005). DOI 10.1145/1071610.1071613
27. Brunetti, J.M., Garcia, R., Auer, S.: 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 10.4018/jswis.2013010101
28. Brunk, S., Heim, P.: tFacet: Hierarchical Faceted Exploration of Semantic Data Using Well-Known Interaction Concepts. In: Proceedings of the International Workshop on Data-Centric Interactions on the Web (DCI 2011), *CEUR Workshop Proceedings*, vol. 817, pp. 31–36. CEUR-WS.org (2011)

29. Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The adaptive web: methods and strategies of web personalization. Springer-Verlag, Berlin, Heidelberg (2007)
30. Bruza, P.D., van der Weide, T.P.: Stratified hypermedia structures for information disclosure. Computer Journal **35**(3), 208–220 (1992). DOI 10.1093/comjnl/35.3.208
31. Burnett, M.M.: Visual Programming. In: J.G. Webster (ed.) Wiley Encyclopedia of Electrical and Electronics Engineering. John Wiley & Sons (1999). DOI 10.1002/047134608X.W1707
32. Burnett, M.M., Baker, M.J.: A Classification System for Visual Programming Languages. Journal of Visual Languages and Computing **5**(3), 287–300 (1994). DOI 10.1006/jvlc.1994.1015
33. Cali, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. Web Semantics: Science, Services and Agents on the World Wide Web **14**, 57–83 (2012). DOI 10.1016/j.websem.2012.03.001
34. Campbell, L.J., Halpin, T.A., Proper, H.A.: Conceptual schemas with abstractions making flat conceptual schemas more comprehensible. Data & Knowledge Engineering **20**(1), 39–85 (1996). DOI 10.1016/0169-023X(96)00005-5
35. Cassino, R., Tucci, M.: Developing usable web interfaces with the aid of automatic verification of their formal specification. Journal of Visual Languages and Computing **22**(2), 140–149 (2011). DOI 10.1016/j.jvlc.2010.12.001
36. Catarci, T.: What happened when database researchers met usability. Information Systems **25**(3), 177–212 (2000). DOI 10.1016/S0306-4379(00)00015-6
37. Catarci, T., Costabile, M.F., Levialdi, S., Batini, C.: Visual query systems for databases: A survey. Journal of Visual Languages and Computing **8**(2), 215–260 (1997). DOI 10.1006/jvlc.1997.0037
38. Catarci, T., Dongilli, P., Di Mascio, T., Franconi, E., Santucci, G., Tessaris, S.: An ontology based visual tool for query formulation support. In: Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004), *Frontiers in Artificial Intelligence and Applications*, vol. 110, pp. 308–312. IOS Press (2004)
39. Certo, L., Galvao, T., Borges, J.: Time Automaton: A visual mechanism for temporal querying. Journal of Visual Languages and Computing **24**(1), 24–36 (2013). DOI 10.1016/j.jvlc.2012.10.001
40. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Proceedings of the 9th annual ACM symposium on Theory of computing (STOC 1977), pp. 77–90. ACM (1977). DOI 10.1145/800105.803397
41. Chen, P.K., Chen, G.D., Liu, B.J.: HVQS: The Hierarchical Visual Query System for Databases. Journal of Visual Languages and Computing **11**(1), 1–26 (2000). DOI 10.1006/jvlc.1999.0140
42. Chin, J.P., Diehl, V.A., Norman, K.L.: Development of an instrument measuring user satisfaction of the human-computer interface. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1988), pp. 213–218. ACM (1988). DOI 10.1145/57167.57203
43. Cinque, L., Levialdi, S., Ferloni, F.: An expert visual query system. Journal of Visual Languages and Computing **2**(2), 101–113 (1991). DOI 10.1016/S1045-926X(05)80025-4
44. Civili, C., Console, M., De Giacomo, G., Lembo, D., Lenzerini, M., Lepore, L., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Santarelli, V., Savo, D.F.: Mastro Studio: Managing Ontology-based Data Access Applications. Proceedings of the VLDB Endowment **6**(12), 1314–1317 (2013)
45. Claussen, J., Kemper, A., Moerkotte, G., Peithner, K., Steinbrunn, M.: Optimization and evaluation of disjunctive queries. IEEE Transactions on Knowledge and Data Engineering **12**(2), 238–260 (2000). DOI 10.1109/69.842265
46. Codd, E.F.: A relational model of data for large shared databanks. Communications of the ACM **13**(6), 377–387 (1970). DOI 10.1145/362384.362685
47. Codd, E.F.: Relational Completeness of Data Base Sublanguages. In: J. Randall (ed.) Data Base Systems, *Courant Computer Science Symposia Series*, vol. 6, pp. 65–98. Prentice-Hall, Englewood Cliffs, NJ (1972)
48. Codd, E.F.: Extending the database relational model to capture more meaning. ACM Transactions on Database Systems **4**(4), 397–434 (1979). DOI 10.1145/320107.320109
49. Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D.: Context is key. Communications of the ACM **48**(3), 49–53 (2005). DOI 10.1145/1047671.1047703
50. Cuff, R.N.: On casual users. International Journal of Man-Machine Studies **12**(2), 163–187 (1980). DOI 10.1016/S0020-7373(80)80016-2
51. Dadzie, A.S., Rowe, M.: Approaches to Visualising Linked Data: A Survey. Semantic Web **2**(2), 89–124 (2011). DOI 10.3233/SW-2011-0037
52. Damljanovic, D., Agatonovic, M., Cunningham, H., Bontcheva, K.: Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. Web Semantics: Science, Services and Agents on the World Wide Web **19**, 1–21 (2013). DOI 10.1016/j.websem.2013.02.002
53. Deutsch, A., Marcus, M., Sui, L., Vianu, V., Zhou, D.: A verifier for interactive, data-driven web applications. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD 2005), pp. 539–550. ACM (2005). DOI 10.1145/1066157.1066219
54. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing **5**(1), 4–7 (2001). DOI 10.1007/s007790170019
55. Dividino, R., Groner, G.: Which of the following SPARQL Queries are Similar? Why? In: Linked Data for Information Extraction (LD4IE 2013), *CEUR Workshop Proceedings*, vol. 1057. CEUR-WS.org (2013)
56. Doan, K., Plaisant, C., Shneiderman, B.: Query Previews in Networked Information Systems. In: Proceedings of the 3rd International Forum on Research and Technology Advances in Digital Libraries (ADL 1996), pp. 120–129. IEEE Computer Society (1996)
57. D'Ulizia, A., Ferri, F., Grifoni, P.: Moving GeoPQL: a pictorial language towards spatio-temporal queries. Geoinformatica **16**(2), 357–389 (2012). DOI 10.1007/s10707-011-0135-6
58. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artificial Intelligence **172**(12-13), 1495–1539 (2008). DOI 10.1016/j.artint.2008.04.002
59. Elmasri, R., Navathe, S.B.: Database Systems: Models, Languages, Design and Application Programming, 6th edn. Pearson Global Edition (2011)
60. Endres-Niggemeyer, B.: The Mashup Ecosystem. In: B. Endres-Niggemeyer (ed.) Semantic Mashups, pp. 1–50. Springer (2013). DOI 10.1007/978-3-642-36403-7_1

61. Epstein, R.G.: The TableTalk Query Language. Journal of Visual Languages and Computing **2**, 115–141 (1991). DOI 10.1016/S1045-926X(05)80026-6
62. Erwig, M.: Xing: a visual XML query language. Journal of Visual Languages and Computing **14**(1), 5–45 (2003). DOI 10.1016/S1045-926X(02)00074-5
63. Fadhil, A., Haarslev, V.: GLOO: A Graphical Query Language for OWL Ontologies. In: Proceedings of the OWL: Experiences and Directions (OWLED 2006), *CEUR Workshop Proceedings*, vol. 216. CEUR-WS.org (2006)
64. Fonseca, F., Martin, J.: Learning the differences between ontologies and conceptual schemas through ontology-driven information systems. Journal of the Association for Information Systems **8**(2), 129–142 (2007)
65. Friedhoff, R.M., Benzon, W.: Visualization: The Second Computer Revolution. Harry N. Abrahams Inc. (1989)
66. Gaines, B.R.: Designing Visual Languages for Description Logics. Journal of Logic, Language and Information **18**(2), 217–250 (2009). DOI 10.1007/s10849-008-9078-1
67. Gallud, J.A., Lozano, M.D., Vanderdonckt, J.: Distributed user interfaces: Usability and collaboration. International Journal of Human-Computer Studies **72**(1), 44 (2014). DOI http://dx.doi.org/10.1016/j.ijhcs.2013.10.006
68. Gersh, J., Lewis, B., Montemayor, J., Piatko, C., Turner, R.: Supporting insight-based information exploration in intelligence analysis. Communications of the ACM **49**(4), 63–68 (2006). DOI 10.1145/1121949.1121984
69. Giese, M., Calvanese, D., Horrocks, I., Ioannidis, Y., Klappi, H., Koubarakis, M., Lenzerini, M., Moller, R., Ozcep, O., Rodriguez Muro, M., Rosati, R., Schlatte, R., Soylu, A., Waaler, A.: Scalable End-user Access to Big Data. In: A. Rajendra (ed.) Big Data Computing. Chapman and Hall/CRC (2013)
70. Giese, M., Soylu, A., Vega-Gorgojo, G., Waaler, A., Haase, P., Jimenez-Ruiz, E., Lanti, D., Rezk, M., Xiao, G., Ozcep, O., Rosati, R.: Optique: Zooming in on Big Data. Computer **48**(3), 60–67 (2015). DOI 10.1109/MC.2015.82
71. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. Journal of Artificial Intelligence Research **31**(1), 157–204 (2008)
72. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: Ontological Engineering. Advanced Information and Knowledge Processing. Springer-Verlag, Berlin, Heidelberg (2004)
73. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The Next Step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web **6**(4), 309–322 (2008). DOI 10.1016/j.websem.2008.05.001
74. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition **5**(2), 199–221 (1993). DOI 10.1006/knac.1993.1008
75. Guarino, N.: Formal ontology and information systems. In: Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS 1998). IOS Press (1998)
76. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: A Visual Query Notation for Linked Data. In: The Semantic Web: ESWC 2015 Satellite Events, *LNCS*, vol. 9341, pp. 387–402. Springer (2015). DOI 10.1007/978-3-319-25639-9_51
77. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, second edn. Morgan Kaufmann, San Francisco (2008)
78. van Ham, F., van Wijk, J.J.: Beamtrees: compact visualization of large hierarchies. Information Visualization **2**(1), 31–39 (2003). DOI 10.1057/palgrave.ivs.9500036
79. Harth, A.: VisiNav: A system for visual search and navigation on web data. Web Semantics: Science, Services and Agents on the World Wide Web **8**(4), 348–354 (2010). DOI 10.1016/j.websem.2010.08.001
80. Harth, A., Kruk, S.R., Decker, S.: Graphical Representation of RDF Queries. In: Proceedings of the 15th International Conference on World Wide Web (WWW 2006), pp. 859–860. ACM (2006). DOI 10.1145/1135777.1135914
81. Hearst, M.A.: Search User Interfaces. Cambridge University Press (2009)
82. Heim, P., Ziegler, J.: Faceted visual exploration of semantic data. In: Proceedings of the 2nd IFIP WG 13.7 Conference on Human-computer Interaction and Visualization (HCIV 2009), *LNCS*, vol. 6431, pp. 58–75. Springer (2011). DOI 10.1007/978-3-642-19641-6_5
83. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. Journal of Systems and Software **84**(2), 301–313 (2011). DOI 10.1016/j.jss.2010.10.025
84. Hogenboom, F., Milea, V., Frasincar, F., Kaymak, U.: RDF-GL: A SPARQL-Based Graphical Query Language for RDF. In: R. Chbeir, Y. Badr, A. Abraham, A.E. Hassanien (eds.) Emergent Web Intelligence: Advanced Information Retrieval, Advanced Information and Knowledge Processing, pp. 87–116. Springer-Verlag (2010). DOI 10.1007/978-1-84996-074-8_4
85. Howe, D., Costanzo, M., Fey, P., Gojobori, T., Hannick, L., Hide, W., Hill, D.P., Kania, R., Schaeffer, M., St Pierre, S., Twigger, S., White, O., Rhee, S.Y.: Big data: The future of biocuration. Nature **455**(7209), 47–50 (2008). DOI 10.1038/455047a
86. Huynh, D.F., Karger, D.R.: Parallax and companion: set-based browsing for the data web. Online (2009). URL `http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf`
87. Huynh, D.F., Karger, D.R., Miller, R.C.: Exhibit: lightweight structured data publishing. In: Proceedings of the 16th International Conference on World Wide Web (WWW 2007), pp. 737–746. ACM (2007). DOI 10.1145/1242572.1242672
88. Ingwersen, P., Järvelin, K.: The turn: Integration of information seeking and retrieval in context. Springer-Verlag, New York (2005)
89. Jagadish, H.V., Chapman, A., Elkiss, A., Jayapandian, M., Li, Y., Nandi, A., Yu, C.: Making database systems usable. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2007), pp. 13–24. ACM (2007). DOI 10.1145/1247480.1247483
90. Jimenez-Ruiz, E., Grau, B.C., Sattler, U., Schneider, T., Berlanga, R.: Safe and economic re-use of ontologies: a logic-based methodology and tool support. In: Proceedings of the 5th European Semantic Web Conference (ESWC 2008), *LNCS*, vol. 5021, pp. 185–199. Springer (2008). DOI 10.1007/978-3-540-68234-9_16
91. Jimeno-Yepes, A., Jiménez-Ruiz, E., Berlanga-Llavori, R., Rebholz-Schuhmann, D.: Reuse of terminological resources for efficient ontological engineering in Life Sciences. BMC Bioinformatics **10**(10), 1–13 (2009). DOI 10.1186/1471-2105-10-S10-S4
92. Kapetanios, E., Baer, D., Groenewoud, P.: Simplifying syntactic and semantic parsing of NL-based queries in advanced application domains. Data & Knowledge Engi-

neering **55**(1), 38–58 (2005). DOI 10.1016/j.datak.2004.11.008
93. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods - A survey. ACM Computing Surveys **39**(4), 10:1–10:43 (2007). DOI 10.1145/1287620.1287621
94. Kaufmann, E., Bernstein, A.: Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. Web Semantics: Science, Services and Agents on the World Wide Web **8**(4), 377–393 (2010). DOI 10.1016/j.websem.2010.06.001
95. Kawash, J.: Complex Quantification in Structured Query Language (SQL): A Tutorial Using Relational Calculus. Journal of Computers in Mathematics and Science Teaching **23**(2), 169–190 (2004)
96. Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., Bilidas, D., Giese, M., Haase, P., Horrocks, I., Kllapi, H., Koubarakis, M., Özçep, O., Rodríguez-Muro, M., Rosati, R., Schmidt, M., Schlatte, R., Soylu, A., Waaler, A.: Optique: Towards OBDA Systems for Industry. In: Proceedings of the Semantic Web: ESWC 2013 Satellite Events, *LNCS*, vol. 7955, pp. 125–140. Springer (2013). DOI 10.1007/978-3-642-41242-4_11
97. Khoussainova, N., Kwon, Y., Liao, W.T., Balazinska, M., Gatterbauer, W., Suciu, D.: Session-based browsing for more effective query reuse. In: Proceedings of the 23rd International Conference on Scientific and Statistical Database Management (SSDBM 2011), *LNCS*, vol. 6809, pp. 583–585. Springer (2011). DOI 10.1007/978-3-642-22351-8_47
98. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. Journal of the ACM **42**(4), 741–843 (1995). DOI 10.1145/210332.210335
99. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In: The Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), *LNCS*, vol. 3298, pp. 229–243. Springer (2004). DOI 10.1007/978-3-540-30475-3_17
100. Kobilarov, G., Dickinson, I.: Humboldt: Exploring Linked Data. In: Proceedings of the Linked Data on the Web Workshop (2008)
101. Kogalovsky, M.R.: Ontology-Based Data Access Systems. Programming and Computer Software **38**(4), 167–182 (2012). DOI 10.1134/S0361768812040032
102. Kolomiyets, O., Moens, M.F.: A Survey on Question Answering Technology from an Information Retrieval Perspective. Information Sciences **181**(24), 5412–5434 (2011). DOI 10.1016/j.ins.2011.07.047
103. Kondylakis, H., Plexousakis, D.: Ontology evolution without tears. Web Semantics: Science, Services and Agents on the World Wide Web **19**, 42–58 (2013). DOI 10.1016/j.websem.2013.01.001
104. Konstan, J.A., Riedl, J.: Recommender systems: from algorithms to user experience. User Modeling and User-Adapted Interaction **22**(1-2), 101–123 (2012). DOI 10.1007/s11257-011-9112-x
105. Koutrika, G., Zadeh, Z.M., Garcia-Molina, H.: Data clouds: summarizing keyword search results over structured data. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT 2009), pp. 391–402. ACM (2009). DOI 10.1145/1516360.1516406
106. Krivov, S., Williams, R., Villa, F.: GrOWL: A Tool for Visualization and Editing of OWL Ontologies. Web Semantics: Science, Services and Agents on the World Wide Web **5**(2), 54–57 (2007). DOI 10.1016/j.websem.2007.03.005
107. Laney, D.: 3D Data Management: Controlling Data Volume, Velocity and Variety. Tech. rep., META Group (2001)
108. Latapy, M., Magnienb, C., Del Vecchioc, N.: Basic notions for the analysis of large two-mode networks. Social Networks **30**(1), 31–48 (2008). DOI 10.1016/j.socnet.2007.04.006
109. Lederman, S., Klatzky, R.: Haptic perception: A tutorial. Attention, Perception, & Psychophysics **71**(7), 1439–1459 (2009). DOI 10.3758/APP.71.7.1439
110. Leone, S., Geel, M., Mueller, C., Norrie, M.C.: Exploiting Tag Clouds for Database Browsing and Querying. In: Proceedings of the Information Systems Evolution - CAiSE Forum 2010, *LNBIP*, vol. 72, pp. 15–28. Springer (2011). DOI 10.1007/978-3-642-17722-4_2
111. Levesque, H.J., Brachman, R.J.: A fundamental tradeoff in knowledge representation and reasoning. In: R.J. Brachman, H.J. Levesque (eds.) Readings in Knowledge Representation, pp. 41–70. Morgan Kaufmann, Los Altos, CA (1985)
112. Lieberman, H., Paterno, F., Wulf, V. (eds.): End User Development, *Human-Computer Interaction Series*, vol. 9. Springer (2006)
113. Lindgaard, G., Dudek, C.: What is this evasive beast we call user satisfaction? Interacting with Computers **15**(3), 429–452 (2003). DOI 10.1016/S0953-5438(02)00063-2
114. Lohse, G.L., Biolsi, K., Walker, N., Rueter, H.H.: A classification of visual representations. Communications of the ACM **37**(12), 36–49 (1994). DOI 10.1145/198366.198376
115. Lopez, V., Unger, C., Cimiano, P., Motta, E.: Evaluating question answering over linked data. Web Semantics: Science, Services and Agents on the World Wide Web **21**, 3–13 (2013)
116. Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Transactions on Graphics **5**(2), 110–141 (1986). DOI 10.1145/22949.22950
117. Madden, S.: From Databases to Big Data. IEEE Internet Computing **16**(3), 4–6 (2012). DOI 10.1109/MIC.2012.50
118. Marchionini, G.: Exploratory search: From finding to understanding. Communications of the ACM **49**(4), 41–46 (2006). DOI 10.1145/1121949.1121979
119. Marchionini, G., White, R.: Find what you need, understand what you find. International Journal of Human-Computer Interaction **23**(3), 205–237 (2007). DOI 10.1080/10447310701702352
120. Martinez-Cruz, C., Blanco, I.J., Amparo Vila, M.: Ontologies versus relational databases: are they so different? A comparison. Artificial Intelligence Review **38**(4), 271–290 (2012). DOI 10.1007/s10462-011-9251-9
121. McAfee, A., Brynjolfsson, E.: Big Data: The Management Revolution. Harvard Business Review **90**(10), 60–68 (2012)
122. Mendes, P.N., Mcknight, B., Sheth, A.P., Kissinger, J.C.: TcruziKB: Enabling Complex Queries for Genomic Data Exploration. In: Proceedings of the IEEE International Conference on Semantic Computing, pp. 432–439. IEEE (2008). DOI 10.1109/ICSC.2008.93
123. Minker, J.: Logic and databases - Fast, present, and future. AI Magazine **18**(3), 21–47 (1997)
124. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation, W3C (2009). URL `http://www.w3.org/TR/owl-profiles/`

125. Motik, B., Rosati, R.: Reconciling description logics and rules. Journal of the ACM **57**(5), 30:1–30:62 (2008). DOI 10.1145/1754399.1754403
126. Munir, K., Odeh, M., McClatchey, R.: Ontology-driven relational query formulation using the semantic and assertional capabilities of OWL-DL. Knowledge-based Systems **35**, 144–159 (2012). DOI 10.1016/j.knosys.2012.04.020
127. Nandi, A., Jagadish, H.V.: Assisted querying using instant-response interfaces. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2007), pp. 1156–1158. ACM (2007). DOI 10.1145/1247480.1247640
128. Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco, CA (1993)
129. Nielsen, J.: Guerrilla HCI: using discount usability engineering to penetrate the intimidation barrier. In: R.G. Bias, D.J. Mayhew (eds.) Cost-justifying usability, pp. 245–272. Academic Press Inc., Orlando, FL, USA (1994)
130. Norman, D.A., Draper, S.W. (eds.): User Centered System Design: New Perspectives on Human-computer Interaction. L. Erlbaum Associates Inc., Hillsdale, NJ (1986)
131. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics (2001)
132. Nunamaker, J.F., Briggs, R.O., de Vreede, G.J.: From Information Technology to Value Creation Technology. In: Information Technology and the Future Enterprise: New Models for Managers, pp. 102–124. Prentice-Hall, New York (2001)
133. Philippi, S.: Model driven generation and testing of object-relational mappings. Journal of Systems and Software **77**(2), 193–207 (2005). DOI 10.1016/j.jss.2004.07.252
134. Pirolli, P., Stuart, C.: The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In: Proceedings of the 2005 International Conference on Intelligence Analysis (2005)
135. Plaisant, C., Grosjean, J., Bederson, B.B.: SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation. In: Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2002), pp. 57–64. IEEE Computer Society (2002). DOI 10.1109/INFVIS.2002.1173148
136. Pollitt, A.S., Smith, M.P., Treglown, M., Braekevelt, P.: View-based searching systems - progress towards effective disintermediation. In: The Proceedings of the Online Information, pp. 433–441 (1996)
137. Popov, I.O., Schraefel, M.C., Hall, W., Shadbolt, N.: Connecting the Dots: A Multi-pivot Approach to Data Exploration. In: Proceedings of the 10th International Semantic Web Conference (ISWC 2011), *LNCS*, vol. 7031, pp. 553–568. Springer (2011). DOI 10.1007/978-3-642-25073-6_35
138. Priyatna, F., Corcho, O., Sequeda, J.: Formalisation and Experiences of R2RML-based SPARQL to SQL query translation using Morph. In: Proceedings of the 23rd International Conference on World Wide Web Conference (WWW 2014) (2014)
139. van Rijsbergen, C.J.: Information Retrieval, 2 edn. Butterworth-Heinemann (1979)
140. Robertson, P.K.: A Methodology for Choosing Data Representations. IEEE Computer Graphics and Applications **11**(3), 56–67 (1991). DOI 10.1109/38.79454
141. Rodriguez-Muro, M., Calvanese, D.: High Performance Query Answering over DL-Lite Ontologies. In: Proceedings of the Principles of Knowledge Representation and Reasoning (KR 2012), pp. 308–318. AAAI Press (2012)
142. Rodriguez-Muro, M., Calvanese, D.: Quest, a System for Ontology Based Data Access. In: Proceedings of the 9th OWL: Experiences and Directions Workshop (OWLED 2012), *CEUR Workshop Proceedings*, vol. 849. CEUR-WS.org (2012)
143. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-Based Data Access: Ontop of Databases. In: Proceedings of the 12th International Semantic Web Conference (ISWC 2013), *LNCS*, vol. 8218, pp. 558–573. Springer (2013). DOI 10.1007/978-3-642-41335-3_35
144. Rodriguez-Muro, M., Lubyte, L., Calvanese, D.: Realizing ontology based data access: A plug-in for Protégé. In: Proceedings of the IEEE 24th International Conference on Data Engineering Workshop (ICDEW 2008), pp. 353–356. IEEE (2008). DOI 10.1109/ICDEW.2008.4498333
145. Ruiz, F., Hilera, J.R.: Using Ontologies in Software Engineering and Technology. In: C. Calero, F. Ruiz, M. Piattini (eds.) Ontologies for Software Engineering and Software Technology, pp. 49–102. Springer-Verlag (2006). DOI 10.1007/3-540-34518-3_2
146. Salehie, M., Tahvildari, L.: Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems **4**(2), 14:1–14:42 (2009). DOI 10.1145/1516533.1516538
147. Schraefel, M.C., Wilson, M., Russell, A., Smith, D.A.: mSpace: improving information access to multimedia domains with multimodal exploratory search. Communications of the ACM **49**(4), 47–49 (2006). DOI 10.1145/1121949.1121980
148. Segev, A., Sheng, Q.Z.: Bootstrapping Ontologies for Web Services. IEEE Transactions on Services Computing **5**(1), 33–44 (2012). DOI 10.1109/TSC.2010.51
149. Sequeda, J.F., Miranker, D.P.: Ultrawrap: SPARQL Execution on Relational Data. Web Semantics: Science, Services and Agents on the World Wide Web **22**, 19–39 (2013). DOI 10.1016/j.websem.2013.08.002
150. Shneiderman, B.: Direct Manipulation: A Step Beyond Programming Languages. Computer **16**(8), 57–69 (1983). DOI 10.1109/MC.1983.1654471
151. Shneiderman, B.: Dynamic queries for visual information seeking. IEEE Software **11**(6), 70–77 (1994). DOI 10.1109/52.329404
152. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: Proceedings of the IEEE Symposium on Visual Languages (VL 1996), pp. 336–343. IEEE Computer Society (1996). DOI 10.1109/VL.1996.545307
153. Siau, K.: A visual object-relationship query language for user-database interaction. Telematics and Informatics **15**(1-2), 103–119 (1998)
154. Siau, K.L., Chan, H.C., Wei, K.K.: Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans **34**(2), 276–281 (2004). DOI 10.1109/TSMCA.2003.820581
155. Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.: A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In: Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns (EKAW 2008), *LNCS*, vol. 5268, pp. 275–291. Springer (2008). DOI 10.1007/978-3-540-87696-0_25

156. Soylu, A., De Causmaecker, P., Desmet, P.: Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering. Journal of Software **4**(9), 992–1013 (2009). DOI 10.4304/jsw.4.9.992-1013
157. Soylu, A., De Causmaecker, P., Preuveneers, D., Berbers, Y., Desmet, P.: 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 10.1504/IJMSO.2011.046595
158. Soylu, A., Giese, M.: Qualifying Ontology-based Visual Query Formulation. In: Proceedings of the Flexible Query Answering Systems (FQAS 2015), *Advances in Intelligent Systems and Computing*, vol. 400, pp. 243–255. Springer (2015). DOI 10.1007/978-3-319-26154-6_19
159. Soylu, A., Giese, M., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: Towards Exploiting Query History for Adaptive Ontology-based Visual Query Formulation. In: Proceedings of the 8th Metadata and Semantics Research Conference (MTSR 2014), CCIS, pp. 107–119. Springer (2014). DOI 10.1007/978-3-319-13674-5_11
160. Soylu, A., Giese, M., Jimenez-Ruiz, E., Vega-Gorgojo, G., Horrocks, I.: 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 10.1007/s10209-015-0404-5
161. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Horrocks, I.: Ontology-based Visual Query Formulation: An Industry Experience. In: Proceedings of the 11th International Symposium on Visual Computing (ISVC 2015), *LNCS*, vol. 9474, pp. 842–854. Springer (2015). DOI 10.1007/978-3-319-27857-5_75
162. Soylu, A., Modritscher, F., De Causmaecker, P.: Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. Integrated Computer-Aided Engineering **19**(1), 93–109 (2012). DOI 10.3233/ICA-2012-0393
163. Soylu, A., Moedritscher, F., Wild, F., De Causmaecker, P., Desmet, P.: Mashups by orchestration and widget-based personal environments: Key challenges, solution strategies, and an application. Program: Electronic Library and Information Systems **46**(4), 383–428 (2012). DOI 10.1109/ICC.2010.5502398
164. Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the Semantic Web: A survey. Semantic Web **3**(2), 169–209 (2012). DOI 10.3233/SW-2011-0055
165. Spiekermann, S.: User Control in Ubiquitous Computing: Design Alternatives and User Acceptance. Shaker Verlag, Aachen, Germany (2008)
166. Staab, S., Studer, R. (eds.): Handbook on Ontologies. International Handbooks on Information Systems. Springer, Berlin, Heidelberg (2009)
167. Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N.W., Goble, C.A., Brass, A.: TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. Bioinformatics **16**(2), 184–186 (2000). DOI 10.1093/bioinformatics/16.2.184
168. Storrle, H.: VMQL: A visual language for ad-hoc model querying. Journal of Visual Languages and Computing **22**(1), 3–29 (2011). DOI 10.1016/j.jvlc.2010.11.004
169. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge Engineering: Principles and methods. Data & Knowledge Engineering **25**(1-2), 161–197 (1998). DOI 10.1016/S0169-023X(97)00056-6
170. Suh, B., Bederson, B.B.: OZONE: a zoomable interface for navigating ontology information. In: Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2002), pp. 139–143. ACM (2002). DOI 10.1145/1556262.1556284
171. Ter Hofstede, A.H.M., Proper, H.A., Van Der Weide, T.P.: Query formulation as an information retrieval problem. Computer Journal **39**(4), 255–274 (1996). DOI 10.1093/comjnl/39.4.255
172. Thompson, C.W., Ross, K.M., Tennant, H.R., Saenz, R.M.: Building Usable Menu-Based Natural Language Interfaces To Databases. In: Proceedings of the 9th International Conference on Very Large Data Bases (VLDB 1983), pp. 43–55. Morgan Kaufmann Publishers Inc. (1983)
173. Tran, T., Herzig, D.M., Ladwig, G.: SemSearchPro - Using semantics throughout the search process. Web Semantics: Science, Services and Agents on the World Wide Web **9**(4), 349–364 (2011). DOI 10.1016/j.websem.2011.08.004
174. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: Live views on the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web **8**(4), 355–364 (2010). DOI 10.1016/j.websem.2010.08.003
175. Tunkelang, D., Marchionini, G.: Faceted Search. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan and Claypool Publishers (2009)
176. Turk, M., Robertson, G.: Perceptual user interfaces (introduction). Communications of the ACM **43**(3), 32–34 (2000). DOI 10.1145/330534.330535
177. Uren, V., Lei, Y., Lopez, V., Liu, H., Motta, E., Giordanino, M.: The usability of semantic search tools: a review. Knowledge Engineering Review **22**(4), 361–377 (2007). DOI 10.1017/S0269888907001233
178. Warschauer, M., Ahumada Newhart, V.: Broadening our concepts of universal access. Universal Access in the Information Society (to appear). DOI 10.1007/s10209-015-0417-0
179. Whang, K.Y., Ammann, A., Bolmarcich, A., Hanrahan, M., Hochgesang, G., Huang, K.T., Khorasani, A., Krishnamurthy, R., Sockut, G., Sweeney, P., Waddle, V., Zloof, M.: Office-by-example: an integrated office system and database manager. ACM Transactions on Information Systems **5**(4), 393–427 (1987). DOI 10.1145/42196.42200
180. White, R.W., Kules, B., Drucker, S.M., Schraefel, M.C.: Supporting exploratory search. Communications of the ACM **49**(4), 37–39 (2006). DOI 10.1145/1121949.1121978
181. Wilson, M.L., Schraefel, M.C.: mSpace: What do numbers and totals mean in a flexible semantic browser. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI 2006) (2006)
182. Wilson, M.L., Schraefel, M.C., White, R.W.: Evaluating Advanced Search Interfaces Using Established Information-Seeking Models. Journal of the American Society for Information Science and Technology **60**(7), 1407–1422 (2009). DOI 10.1002/asi.21080
183. Yang, Y., Wu, X., Zhu, X.: Combining Proactive and Reactive Predictions for Data Streams. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KD 2005), pp. 710–715. ACM (2005). DOI 10.1145/1081870.1081961
184. Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: Proceedings of the SIGCHI Conference on Human Factors in Com-

puting Systems (CHI 2003), pp. 401–408. ACM (2003). DOI 10.1145/642611.642681

185. Yen, M.Y.M., Scamell, R.W.: A Human Factors Experimental Comparison of SQL and QBE. IEEE Transactions on Software Engineering **19**(4), 390–409 (1993). DOI 10.1109/32.223806
186. Yu, C., Jagadish, H.V.: Schema summarization. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), pp. 319–330. VLDB Endowment (2006)
187. Zhang, J., Marchionini, G.: Evaluation and evolution of a browse and search interface: Relation Browser++. In: Proceedings of the 2005 National Conference on Digital Government Research (dg.o 2005), pp. 179–188. Digital Government Society of North America (2005)
188. Zheng, K., Mei, Q., Hanauer, D.A.: Collaborative search in electronic health records. Journal of the American Medical Informatics Association **18**(3), 282–291 (2011). DOI 10.1136/amiajnl-2011-000009
189. Zloof, M.M.: Query-by-example: a database language. IBM System Journal **16**(4), 324–343 (1997). DOI 10.1147/sj.164.0324
190. Zviedris, M., Barzdins, G.: ViziQuer: a tool to explore and query SPARQL endpoints. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011), *LNCS*, vol. 6644, pp. 441–445. Springer (2011). DOI 10.1007/978-3-642-21064-8_31