



# Unifying Faceted Search and Analytics over RDF Knowledge Graphs

Maria-Evangelia Papadaki<sup>1,2</sup> · Yannis Tzitzikas<sup>1,2</sup>

Received: 23 June 2023 / Revised: 15 October 2023 / Accepted: 8 February 2024 /

Published online: 24 March 2024

© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2024

## Abstract

The formulation of analytical queries over Knowledge Graphs in RDF is a challenging task that presupposes familiarity with the syntax of the corresponding query languages and the contents of the graph. To alleviate this problem, we introduce a model for aiding users in formulating analytic queries over complex, i.e., not necessarily star schema-based, RDF Knowledge Graphs. To come up with an intuitive interface, we leverage the familiarity of users with Faceted Search systems. In particular, we start from a general model for Faceted Search over RDF data, and we extend it with actions that enable users to formulate analytic queries, too. Thus, the proposed model can be used not only for formulating analytic queries but also for exploratory purposes, i.e., for locating the desired resources in a Faceted Search manner. We describe the model from various perspectives, i.e., (1) we propose a generic user interface for intuitively analyzing RDF Knowledge Graphs, (2) we define formally the state space of the interaction model and the required algorithms for producing the user interface actions, (3) we present an implementation of the model that showcases its feasibility, and (4) we discuss the results of an evaluation with users that provides evidence for the acceptance of the method by users. Apart from being intuitive for end users, another distinctive characteristic of the proposed model is that it allows the gradual formulation of complex analytic queries (including nested ones).

**Keywords** Knowledge Graphs · Analytics · Faceted Search

## 1 Introduction

There are several Knowledge Graphs (KGs) expressed in RDF (Resource Description Framework) that integrate data from various sources, from *general purpose* KGs (like DBpedia [1])

---

✉ Maria-Evangelia Papadaki  
marpap@ics.forth.gr

Yannis Tzitzikas  
tzitzik@ics.forth.gr

<sup>1</sup> Institute of Computer Science, FORTH-ICS, Vasilika Vouton, 70013 Heraklion, Crete, Greece

<sup>2</sup> Department of Computer Science, University of Crete, Panepistimioupoli Vouton, 70013 Heraklion, Crete, Greece

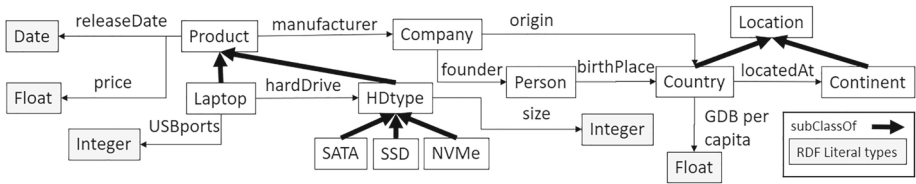


Fig. 1 Schema of the running example

and Wikidata [2]) to *domain specific* semantic repositories, like Europeana [3, 4] for the cultural domain, DrugBank [5] for drugs, GRSF [6] for the marine domain, WarSampo [7] for historical data, ORKG [8] for scholarly works, and [9–11] for COVID-19-related datasets. There are also Markup data through `schema.org` expressed in RDF, as well as Knowledge Graphs producible from plain file systems [12]. Finally, Knowledge Graphs recently are used also for validating and enriching the responses of Large Language Models (like ChatGPT), for example in [13].

Plain users can (1) *browse* such graphs (i.e., the user can start from a resource, inspect its values, and move to a connected resource, and so on, or even decide to move to the more similar resources, e.g., [14]), (2) *search* them using *keyword search* where the emphasis is on the ranking of the resources according to their relevance to the submitted query (e.g., see the multi-perspective keyword search approach described in [15]), or (3) use *interactive query formulators* that aim at aiding the user to formulate a structured query (like A-QuB [16], FedViz [17], SPARKLIS [18], and SPARQL-QBE [19]). However, the latter, i.e., *structured query formulation*, is in general difficult for ordinary users, and it seems that there is no standard, or widely accepted, method of such query formulators, especially for *analytic queries*. To this end, in this paper, we focus on the formulation of *analytic queries* over RDF graphs, a task that for ordinary users is considered almost infeasible, while for expert users it is laborious and time-consuming. We aim at providing a user-friendly method that will allow even novice users to formulate analytic queries over Knowledge Graphs, easily and intuitively.

To emphasize that need, consider a KG with information about products and related entities (companies, persons, locations, etc.) with schema as shown in Fig. 1 (for reasons of brevity namespaces are not shown). Suppose that we want to find “*the average price of laptops made in 2023 from US companies that have 2 USB ports and an SSD drive manufactured in Asia grouped by manufacturer*”. This information need can be expressed in SPARQL as shown in Fig. 2. Obviously, the formulation of such queries is quite difficult for novice users who do not have the required technical background.

Consequently, there is a need for an interaction model that will let users formulate such analytic queries with simple clicks, without presupposing knowledge neither of the vocabulary (schema, ontology, thesauri), nor the actual contents of the dataset, nor the syntax of the corresponding query language (i.e., SPARQL). To this end, we leverage the familiarity of users with *Faceted Search* [20], since this model supports the expression of complex conditions with simple clicks. In particular, we start from a general model for Faceted Search over RDF data, specifically from the core model for faceted exploration of RDF data (described in [21]), and we extend it with actions that enable users to formulate analytic queries, too. The user actions are automatically translated to a query according to the high-level query language for analytics, called HIFUN, and then, the HIFUN query is translated to a SPARQL query (see also Fig. 3). In brief, the contributions of this paper are: (1) we propose a generic

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix      ex:<http://www.ics.forth.gr/example#>
SELECT ?m (AVG(?p) as ?avgprice)
WHERE {
?s rdf:type ex:Laptop.
?s ex:manufacturer ?m.
?m ex:origin ex:USA.
?s ex:price ?p.
?s ex:USBPorts ?u.
?s ex:hardDrive ?hd.
?hd rdf:type ex:SSD.
?hd ex:manufacturer ?hdm.
?hdm ex:origin ?hdmc.
?hdmc ex:locatedAt ex:Asia.
FILTER (?u >= 2).
?s ex:releaseDate ?rd .
FILTER ( ?rd >= "2023-01-01T00:00:00"^^xsd:dateTime &&
?rd <= "2023-12-31T00:00:00"^^xsd:dateTime)
} GROUP BY ?m

```

**Fig. 2** Expression in SPARQL of the query “average price of laptops made in 2021 from US companies that have 2 USB ports and an SSD drive manufactured in Asia grouped by manufacturer”

user interface for intuitively analyzing RDF Knowledge Graphs without pre-supposing any technical knowledge, (2) we define formally the state space of the interaction model and the required algorithms for producing a UI (user interface) that supports this state space, (3) we present an implementation of the model that showcases its feasibility, and (4) we discuss the results of an evaluation of the proposed system with users.

The distinctive characteristics of the proposed model are: (1) it can be applied to any RDF dataset (i.e., not only to star-schema datasets), (2) it provides *guidance* in the sense that the generated SPARQL queries never produce empty results, and it provides *count* information during the interaction, (3) it offers *expressive power* in the sense that it supports arbitrarily long *paths*; it supports the formulation of HAVING clauses, as well as *nested* analytic queries, and (4) it supports both Faceted Search and analytic queries.

The basic idea was demonstrated in the demo paper [22]. In this paper, we detail and further elaborate on this direction, specifically: (1) we discuss in more detail the related works and our placement, (2) we formally specify the interaction model with *states* and *transitions*, (3) we express the query requirements of the model formally using a query language-independent formalism (HIFUN) facilitating in this way the implementation of the model over different technologies, query languages, and triplestores; (4) we provide the exact specification of the UI and the algorithms for facilitating the implementation of the model, (5) we describe the implementation of the model and its evaluation from various perspectives.

The rest of this paper is organized as follows: Sect. 2 provides the required background. Section 3 describes related work. Section 4 describes the basic idea behind interaction model including the GUI extensions. Section 5 discusses the extensions (in comparison to FS) that

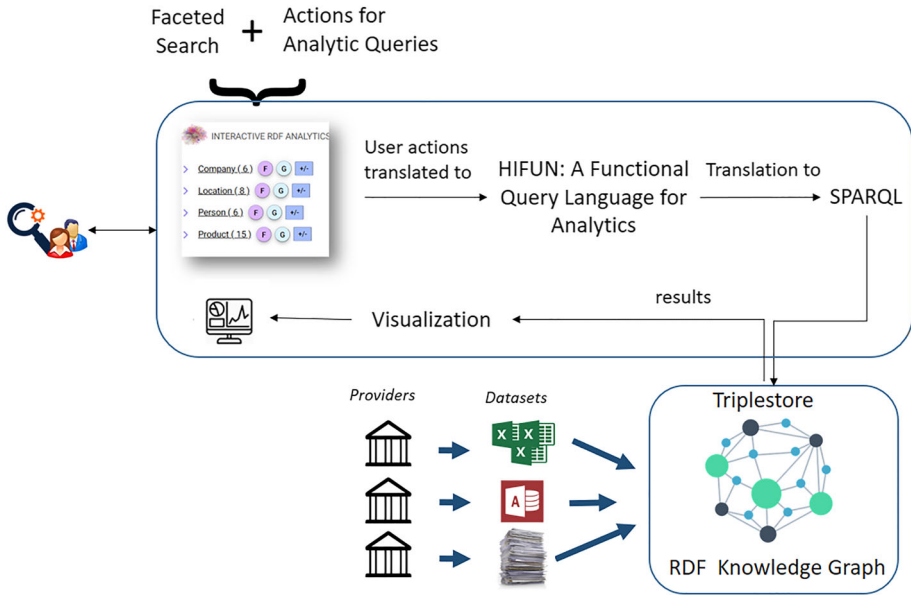


Fig. 3 Context and main elements

are required for analytics. Section 6 provides the formal notations and key points of the algorithms, Sect. 7 presents the algorithm that computes the state space, and Sect. 8 focuses on how the intentions of the states are expressed and computed. Section 9 discusses an implementation of the model. Section 10 describes the expressive power of the model. Section 11 discusses evaluation. Finally, Sect. 12 concludes the paper and identifies issues for further research.

## 2 Background

Here we briefly discuss RDF (in Sect. 2.1), Faceted Search (in Sect. 2.2), and HIFUN (in Sect. 2.3). A diagram that illustrates how these are connected (and the general context) is given in Fig. 3.

### 2.1 The Resource Description Framework (RDF)

**Resource Description Framework (RDF)** The Resource Description Framework (RDF) [23, 24] is a graph-based data model for linked data interchanging on the web. It uses triples, i.e., statements of the form *subject – predicate – object*, where the *subject* corresponds to an entity (e.g., a product, a company etc.), the *predicate* to a characteristic of the entity (e.g., price of a product, location of a company), and the *object* to the value of the predicate for the specific subject (e.g., “300,” “US”). Formally, a *triple* is considered to be any element of  $T = (U \cup B) \times (U) \times (U \cup B \cup L)$ , where  $U$ ,  $B$  and  $L$  denote the sets of URIs, blank nodes and literals, respectively. Any finite subset of  $T$  constitute an *RDF graph* (or *RDF dataset*).

**RDF Schema.** RDF Schema<sup>1</sup> is a special vocabulary that enables the definition of schemas that can be used for describing the resources in a more expressive, semantic, way. RDF Schema enables defining *classes* that can be used for categorizing the resources. It also enables defining *properties* to build relationships between the entities of a class and to model constraints. A class  $C$  is defined by a triple of the form  $\langle C \text{ rdf:type rdfs:Class} \rangle$  using the predefined class “rdfs:Class” and the predefined property “rdf:type.” For example, the triple  $\langle \text{ex:Product rdf:type rdfs:Class} \rangle$  indicates that “Product” is a class, while the triple  $\langle \text{ex:product1 rdf:type ex:Product} \rangle$  that the individual “*product1*” is an instance of class *Product*. A property can be defined by stating that it is an instance of the predefined class “rdf:Property.” Optionally, properties can be declared to be applied to certain instances of classes by defining their *domain* and *range* using the predicates “rdfs:domain” and “rdfs:range,” respectively. For example, the triples  $\langle \text{ex:manufacturer rdf:type rdf:Property} \rangle$ ,  $\langle \text{ex:manufacturer rdfs:domain ex:Product} \rangle$ ,  $\langle \text{ex:manufacturer rdfs:range ex:Company} \rangle$  indicate that the domain of the property “manufacturer” is the class “Product” and its range the class “Company.” The RDF Schema is also used for defining *hierarchical relationships* among classes and properties. The predefined property “rdfs:subclassOf” is used as a predicate in a statement to declare that a class is a specialization of another more general class, while the specialization relationship between two properties is described using the predefined property “rdfs:subPropertyOf.” For example, the triple  $\langle \text{ex:Laptop rdfs:subClassOf ex:Product} \rangle$  denotes that the class “Laptop” is sub-class of the “Product” class. And the triple  $\langle \text{ex:fatherOf rdf:subPropertyOf ex:parentOf} \rangle$  defines that the property “fatherOf” is sub-property of “parentOf.” In addition, RDFS offers *inference* functionality,<sup>2</sup> for example, if  $\langle \text{ex:myLaptop rdf:type ex:Laptop} \rangle$  and  $\langle \text{ex:Laptop rdf:subClassOf ex:Product} \rangle$ , then it can be deduced that “*ex:myLaptop rdf:type ex:Product*,” i.e., that *ex:myLaptop* is also a Product.

## 2.2 Faceted Search

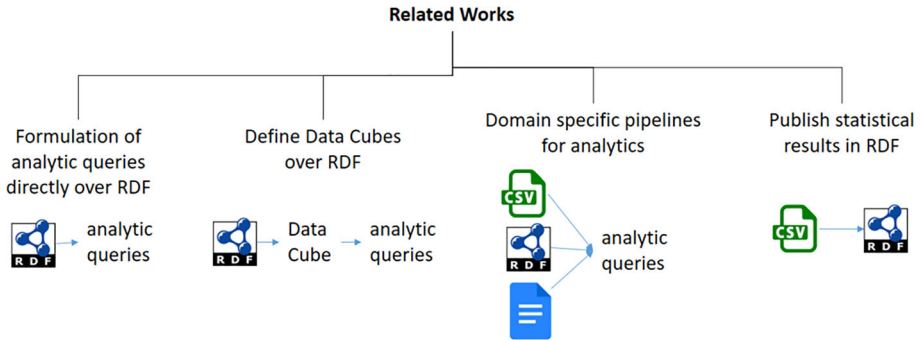
**Faceted Search** (or *Faceted Exploration*) [25–27] is a widely used interaction scheme for Exploratory Search. It is the de facto query paradigm in e-commerce [20, 28, 29] and in digital libraries [30, 31]. It is also used for exploring RDF Data (e.g., see [21] for a recent survey, and [32, 33] for recent systems), as well as for exploring general purpose Knowledge Graphs [34, 35]. There are also recent extensions of the model, with *preferences* and *answer size constraints* [33]. Informally we could define Faceted Search as a *session-based interactive method for query formulation (commonly over a multidimensional information space) through simple clicks that offers an overview of the result set (groups and count information), never leading to empty result sets.*

## 2.3 HIFUN: a functional query language for analytics

HIFUN [36] is a high-level functional query language for defining analytic queries over big datasets, independently of how these queries are evaluated. It can be applied over a dataset that is structured or unstructured, homogeneous or heterogeneous, centrally stored, or distributed. To apply that language over a dataset  $D$ , two assumptions should hold: The dataset should (1) consist of *uniquely identified data items* and (2) have a set of *attributes* each of which is

<sup>1</sup> [https://en.wikipedia.org/wiki/RDF\\_Schema](https://en.wikipedia.org/wiki/RDF_Schema).

<sup>2</sup> <https://www.w3.org/standards/semanticweb/inference>.



**Fig. 4** Spectrum of related works

viewed as a *function* associating each data item of  $D$  with a value, in some set of values. Let  $D$  be a dataset and  $A$  be the set of all attributes ( $a_1, \dots, a_k$ ) of  $D$ . An *analysis context* over  $D$  is any set of attributes from  $A$ , and  $D$  is considered the *origin* (or *root*) of that context. A *query* in HIFUN is defined as an ordered triple  $Q = (g, m, op)$  such that  $g$  and  $m$  are attributes of the dataset  $D$  having a common source and  $op$  is an aggregate operation (or *reduction operation*) applicable on  $m$ -values. The first component of the triple is called *grouping function*, the second *measuring function* (or the *measure*), and the third *aggregate operation* (or *reduction operation*). In addition, one can restrict the three components  $g, m, op$  of a HIFUN query. Thus, the general form of a HIFUN query is  $q = (gE/rg, mE/rm, opE/ro)$ , where  $gE$  is the grouping expression,  $mE$  the measuring expression, and  $opE$  the operation expression, whereas  $rg$  is a restriction on the grouping expression,  $rm$  is a restriction on the measuring expression,  $ro$  is a restriction on the operation expression.

Later, when we will define formally the interaction model, we shall use HIFUN for expressing the formulated analytic query. This enables the exploitation of the model in other contexts where HIFUN is applicable. Returning to Knowledge Graphs over RDF, HIFUN queries are translated to SPARQL queries.

### 3 Related work

*General Positioning and Focus.* We focus on developing a user-friendly interface, where the user will be able to apply analytics to any RDF data in a familiar and gradual manner, without having to be aware of the contents of the dataset(s), nor the lower-level technicalities of SPARQL.

Below we discuss in brief the spectrum of related works, also illustrated in Fig. 4. For a more detailed survey, see [37]. In particular, we describe works about formulating analytic queries directly over RDF (in Sect. 3.1), defining Data Cubes over RDF data (in Sect. 3.2), defining domain-specific pipelines that produce RDF data and support particular analytic queries (in Sect. 3.3) and publishing statistical data in RDF (in Sect. 3.4). In addition, we compare with Query-by-Example (in Sect. 3.5), and finally, we describe our positioning and contribution (in Sect. 3.6).

### 3.1 Formulation of analytic queries directly over RDF

*Formulating Analytics Queries directly over RDF.* Here, we refer to the three most related systems and works, i.e., to work/systems that enable the formulation of analytic queries directly over RDF. At first, [38] proposes an approach for guided query building that supports analytical queries, which can be applied over any RDF graph. The implementation is over the SPARKLIS editor [39], and it has been adopted in a national French project.<sup>3</sup> The approach is interesting; however, during query formulation, no count information is provided, reducing the exploratory characteristics of the process. As regards expressiveness, HAVING-restrictions are not supported. In addition, the GUI is not the classical of FS, so not every user is familiar with it. Nevertheless, the authors report positive evaluation results as regards the expressive power of the interactive formulator. The second is [40] that describes a possible extension of SemFacet [41] to support numeric value ranges and aggregation. The authors also state that they work toward extending the proposed approach to support reachability, too. However, the focus is on theoretical query management aspects, related to Faceted Search, and an interface as well as an implementation is lacked. From the mockups of the GUI, it seems that no count information is provided (which is considered important enough for exploration purposes). Moreover, explicit path expansion is not supported. The authors use the notion of “recursion” to capture reachability-based facet restrictions. Finally, the third one [42] (and the more recent paper [43]) explores the integration of Faceted Search and Visualization as a means to fetch data from SPARQL endpoints and analyze it, effectively. The approach incorporates filters, property paths, and count information. However, the analytical queries formulated within this framework are constrained to basic exploration capabilities Faceted Search offers, lacking the flexibility to impose restrictions on final results. Additionally, no evaluation of the proposed approach has been conducted to validate its effectiveness and performance.

### 3.2 Definition of data cubes over RDF

Another related topic is the *definition of a data cube over RDF*. There are works that implicitly define a data cube over existing RDF graphs<sup>4</sup> [44–46], and then apply OLAP (Online Analytical Processing). One weakness of this approach (as stressed also in [38]) is that it requires someone with technical knowledge to define the required data cube(s). Apart from reduced flexibility, the wealth of connections of the knowledge graph cannot be leveraged, since the user is restricted on one data cube. Also, it is not guaranteed that the constructed RDF Data Cubes will be multi-dimensional compliant and how multi-valued attributes, or empty values, will be treated. Towards this general direction, [47] analyzed the applicability of HIFUN (as described in Sect. 2.3), over RDF. Specifically it provides various methods for defining an *analysis context* (which is analogous to a data cube), and including feature construction operators for cases where the RDF data cannot fit to a cube. Moreover, that work provides the algorithms for translating HIFUN queries to SPARQL queries. What is missing from that work is the interactive formulation of a HIFUN query. In the current paper we want to fill this gap, i.e., we focus on how to formulate the HIFUN query interactively, while exploring the dataset. This task is not easy for users, since it is laborious to find and select the right property from a big schema, let alone the formulation of restrictions. Note that, in small star schema the tasks of selection and restriction formulation are not difficult. How-

<sup>3</sup> <http://data.persee.fr/explore/sparklis/?lang=en>.

<sup>4</sup> <https://team.inria.fr/oak/projects/warg/>.

ever, in knowledge graphs with broad coverage, these tasks can be very laborious. Therefore, methods that can reduce this effort and support exploration are required.

### 3.3 Domain-specific pipelines produce and analyze RDF data

There are also *domain-specific works* (focusing on a particular topic, not on any RDF dataset), like [48] that motivates Knowledge Graph-enabled cancer data analytics. An analogous work for covid-19-related data is [49]. Such works focus on defining specific pipelines for constructing the desired Knowledge Graph and then enabling particular analytic queries and visualizations to support domain-specific research purposes, i.e., they do not provide general-purpose methods for Knowledge Graph analytics.

### 3.4 Publishing of statistical data in RDF

Another related topic is the *publishing of statistical data*. Indeed, there are methods (e.g., [50]) for publishing statistical data as linked data, and toward this end the RDF data cube vocabulary<sup>5</sup> (QB) provides a means to publish such data on the web using the W3C RDF standard. In our work, we do not focus on publishing statistical data, but on formulating analytic queries over any RDF dataset.

### 3.5 Comparison with Query-By-Example

Query-By-Example, for short QBE, refers to a paradigm for interactive query formulation that aims at enabling even non-programmers to query relational databases (e.g., MS ACCESS), and this is achieved by asking the user to provide *examples* of the desired answer. The idea was first proposed for relational databases [51]; however, the main idea can be transferred also to Knowledge Graphs, in particular for aiding the formulation of SPARQL queries. Indeed, there are some works about QBE over KGs, including [19, 52–54].

As regards the comparison between FS (Faceted Search) and QBE, we could say that if the user knows that he wants, i.e., if he knows the desired conditions related to his information need, then FS is beneficial, because it enables him to apply the desired filter conditions(s), during the exploration of the information space. If on the other hand, the user has no idea about the desired constraints, e.g., suppose a user that likes some particular movies but he has no idea about the characteristics of these movies (actors, director, genre, etc.), then a QBE approach is beneficial, since it enables him to use these movies as examples and let the system formulate the constraints that describe these movies. We could therefore say that FS and QBE are complementary approaches. Finally, we should mention that FS is widely used by all users (for e-commerce, bookings, etc.); however, QBE is mainly for database developers. Furthermore, QBE for Knowledge Graphs is still subject of research and has not yet reached all users.

<sup>5</sup> <https://www.w3.org/TR/vocab-data-cube/>.



### 3.6 Our position and contribution

In general, we observe that there are not so many works, neither running systems, that support the easy and intuitive analysis of RDF Knowledge Graphs; consequently, we could say that the majority of RDF datasets are not easily explored and queried.



Our work falls in the category of Sect. 3.1. In Table 1, we qualify the more relevant systems, mentioned in Sect. 3.1, according to some important functionalities, i.e., applicability (if they can be applied over star schemas or over any RDF graph), support of basic analytic queries, support of analytic queries with HAVING clause, support of plain Faceted Search, support of property paths in Faceted Search and analytics, support of results' visualization, offer of running systems, and conduction of an evaluation.


We observe that in comparison with the related systems, our approach (and the system *RDF-ANALYTICS* that we shall present later) out-stands since (1) it can be applied to any RDF graph (not only to star-schema graphs) without requiring data pre-processing, (2) it supports plain browsing of the graph as well as analysis of it, (3) it supports restrictions of the results and property paths, (4) it offers visualization of the results in a running system that is publicly available, and (5) it has been evaluated by users. We should also stress the support of nested queries, as well as the user guidance (stemming from the characteristics of Faceted Search).

As regards comparison with OLAP (Online Analytical Processing) operations, they are described after the presentation of the model, specifically in Sect. 10.2.

Below we describe the proposed approach from three perspectives: (a) we formally specify the interaction model with *states* and *transitions*, (b) we express the query requirements of the model formally using a query language independent formalism facilitating in this way the implementation of the model over different technologies, query languages, and triplestores (see [55] for a survey of triplestores), and (c) we provide the exact specification of the UI and the algorithms followed for facilitating the implementation of the model.

## 4 The interaction model in brief

**GUI Extensions in Brief.** The classical FS interface usually comprises two main frames: the left is used for presenting the facets and the right for displaying the objects, see Fig. 5 (left). The model, that we propose, extends the user actions of the left frame with actions required for the formulation of analytical queries. Specifically, it is enriched with two buttons, i.e.,  and  next to each facet, as shown in Fig. 5 (right). Moreover, an additional frame, called "Answer Frame" (for short *AF*), has been added, for displaying the results of the analytic queries in tabular or graphical format. To grasp the idea, we describe below four (4) indicative examples that show how a simple or complex analytic query can be formulated using our model. We assume that the data of the examples follow the schema of Fig. 1.

**Example 1 (an AVG query without GROUP BY).** Suppose that we would like to find "the average price of laptops made in 2021 from US companies and have SSD and 2 USB ports". The part of the query that refers to specific laptops (i.e., laptops made in 2021 from US companies that have SSD and 2 USB ports) could be expressed by using the classical FS system. Note that the condition "US companies" would be specified by expanding the path of the property "manufacturer" till the property of "origin." What is missing is the specification of the aggregate function (in this case "AVG"). For that, the  button laid on the right on the "price" facet is offered letting the users click and select the desired function from the displayed menu.

**Table 1** Comparing the functionalities of related systems

System	Applicability (STAR vs ANY)	Analytic queries: basic	Analytic queries: with having	Plain Faceted Search	Property paths (in Faceted Search and analytics)	Visualization	Running system	Evaluation
[40]	ANY	Yes	Yes	Yes but with No Count information	Not explicitly, reachability	No	No	No
[38]	ANY	Yes	No	No. Special interface	Not clear	No	Yes	Yes
[42]	ANY	Yes	No	Yes	Yes with counts	Yes	Yes	No
Our approach	ANY	Yes	Yes by AF	Yes	Yes with counts	Yes	Yes	Yes

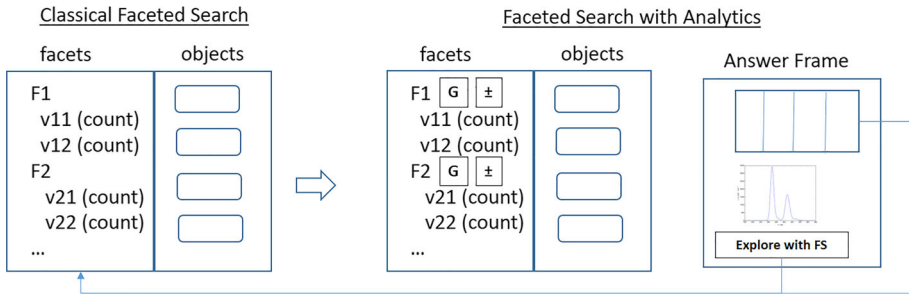


Fig. 5 Core elements of the GUI for Faceted Search and Analytics

**Example 2 (a COUNT query with GROUP BY).** Suppose now that the user would like to find “the count of laptops that made in 2021 and have SSD and 2 USB ports grouped by manufacturers’ country.” The part of the query that refers to specific laptops, i.e., “laptops made in 2021 that have SSD and 2 USB ports,” could be expressed by using the classical FS system. As it has already been mentioned in the previous example, the aggregate function (in this case “count”) would be defined by clicking on the  $\pm$  button laid on the right of the “price” facet and selecting the desired function from the displayed menu. What is missing is the specification of the grouping condition. For that, the  $G$  button laid on the right of each facet lets the users group the results on any set of facets. In this case, the user would click on the  $G$  button laid next to the “origin” facet. Note again that the condition “manufacturers’ country” would be specified by expanding the path of the property “manufacturer” till the property of “origin.”

**Example 3 (a query with range values).** Suppose now that the user would like to find “the count of laptops that made in 2021 and have SSD and 2 or more USB ports grouped by manufacturers’ country.” The only difference with the previous query is that the user should specify the range of the values that refer to USB ports. For that, the  $\square$  button laid on the right of each facet enables the users to filter the values based on the desired range. In this case, the user would click on the  $\square$  button laid next to the “USB ports” facet and (s)he would specify the desired range in the provided form.

**Example 4 (a query with restriction on groups, i.e., with HAVING).** Suppose now that, we would like to find “the average price of laptops grouped by company and year, only for the laptops that have average price above a threshold  $t$ .” The aggregate function and the grouping of the results would be specified via the  $G$  and  $\pm$  buttons laid next to the desired facets, as described in the previous examples. What is missing is to restrict the results over the aforementioned threshold. For that, the answer of an analytical query can be loaded as a new dataset on the extended FS system letting the users further restrict its values. For example, suppose that the results of the first part of the query “average price of laptops grouped by company and year” correspond to the table shown in Fig. 6a. In order to further restrict the answer, we have attached a button, called “Explore with FS” below this table (as shown in Fig. 5) which lets the users load the results as a new dataset on the FS system as shown in Fig. 6b. As it is shown, each column of the table corresponds to a facet of the system having instances the values of the corresponding column. Now, the users can express the desired restriction over the average price by clicking on the “filter” button laid next to the “price” facet and specifying the intended range on the pop-up window that is displayed.








**Fig. 6** Example 3: Exploring the results of an analytic query with Faceted Search

(a)	(b)
DELL 2020 900	Manufacturers
ACER 2021 820	DELL (2)
DELL 2021 1000	ACER (2)
ACER 2021 850	Years
	2020 (2)
	2021 (2)
	Avg Prices
	820 (1)
	850 (1)
	900 (1)
	1000 (1)

**Expressing the Queries of the Previous Examples in HIFUN.** Above we described the interaction. During the interaction, HIFUN queries are formulated. For reasons of completeness, below we show the HIFUN queries that correspond to the aforementioned examples.

- Example 1:  $(\epsilon, price/E, AVG)$ , where  $\epsilon$  is an empty grouping function and  $E = \{i \in D / releaseDate(i) = 2021 \wedge origin \circ manufacturer(i) = US \wedge SSD = true \wedge USBPorts(i) = 2\}$
- Example 2:  $(g/E, ID, COUNT)$ , where  $g = origin \circ manufacturer$ ,  $E = \{i \in D / releaseDate(i) = 2021 \wedge origin \circ manufacturer(i) = US \wedge SSD = true \wedge USBPorts(i) = 2\}$  and  $ID$  is the identity function
- Example 3:  $(g/E, ID, COUNT)$ , where  $g = origin \circ manufacturer$ ,  $E = \{i \in D / year \circ releaseDate(i) = 2021 \wedge origin \circ manufacturer(i) = US \wedge SSD = true \wedge USBPorts(i) \geq 2\}$  and  $ID$  is the identity function
- Example 4:  $(g/E, price, AVG)/F$ , where  $g = (manufacturer \times (\{i \mid i \in D \wedge year \circ releaseDate(i)\}))$ ,  $E = \{i \in D / releaseDate(i) = 2021 \wedge origin \circ manufacturer(i) = US \wedge SSD = true \wedge USBPorts(i) = 2\}$  and  $F = \{g_i \in (g/E) / ans(g_i/E) \geq t\}$

**GUI Extensions.** Below, we describe in brief the UI extensions of the classical FS system that we introduce for supporting analytics, too.

- **Facets:** On the right side of each facet name, there are two buttons: (1) the  button for grouping the results of the analytical query on this facet and (2) the  button that lets the users select the function, i.e., avg, min, max, etc., that will be applied to each group of the analytic results,
- **States of  and  buttons.** If the user clicks on the  button of more than one facets, then the system asks if (s)he wants to group the results by > 1 attributes, or if (s)he wants to remove some of them. Analogously, for : If the user clicks on the  button of more than one facets, then the system asks if (s)he wants to apply > 1 aggregate functions to each group of the analytic results or if (s)he wants to remove some of them.
- **Answer Frame.** A frame is used for showing the results of the analytic query in tabular or graphical format.
- **Michelanea. Extra Columns.** The answer frame could let the users add or remove columns corresponding to the grouping attributes (i.e., display or remove attribute that corresponds to the grouping of the results).

**Special cases.** As stated in [47], there are cases where HIFUN is not applicable on a dataset, i.e., if multi-valued attributes or empty values exist. For those cases, we could add one more

buttons next to each facet name, say  $\boxplus$ , that would let users apply *transformation* functions, i.e., the *feature constructor operators* like those proposed in [47], over it. In addition, such a button would be useful for defining *derived* attributes, e.g., for decomposing a date-based attribute to Year, Month, Day, etc.

## 5 The required extensions of the formal model for FS over RDF for supporting analytics

Here, we describe in brief the basics of the underlying core model for FS over RDF data (in Sect. 5.1) and the required extensions of that model for supporting analytics (in Sect. 5.2).

### 5.1 Background: the core model for FS over RDF

Our approach is based on the general model for Faceted Search over RDF described in [21]. In brief, that model defines the state space of the interaction, where each *state* has an *intention* (query), an *extension* (set of resources), and *transitions* between the states. Each transition is enacted through a *transition marker* (anywhere that the user can click). The approach is generic in the sense that it is independent from the particular query languages (QLs) (used for expressing the intentions of the model).

That work describes formally how the transitions of a given state are determined and how each click is interpreted, i.e., how the new state is generated, etc. Distinctive characteristics of the model presented, in comparison with the classical FS system, are that: (1) it leverages the `rdfs:subClassOf` and `rdfs:subPropertyOf` relations, (2) it supports the formulation of path expressions (exploiting the RDF principles), and (3) it supports switching of entity types. Thus, this model leverages the complexity of RDF graphs.

### 5.2 The extension of the model for analytics (formally)

Let *Obj* be the set of all objects (i.e., all individuals in our case), let *ctx* the current state, *ctx.Ext* be the set of objects of the current focus (i.e., displayed objects), and *ctx.Int* be a query whose answer is *ctx.Ext*. The question is: How a HIFUN query can be formulated over such an FS system?

Initially, the user should specify the context of analysis, this is, the set of object in *ctx.Ext*, and the attributes to be analyzed, which are the properties applicable to *ctx.Ext*. Next, (s)he should specify the grouping function, the measuring function and the aggregate operation of the query. Recall that the general form of a HIFUN query is  $q = (gE/rg, mE/rm, opE/ro)$ . Each of these parts of the query can be specified through the  $\boxminus$  and  $\boxplus$  buttons laid next to each facet. In particular,

- clicking on  $f.\boxminus$ : when the user click on  $f.\boxminus$  the intention *ctx.Int* of the model is changed. Specifically, the grouping function is defined as:  $gE' = gE + f$ , where *f* can denote a facet or a property path. If *f* corresponds to a value, then a restriction on the grouping function is applied.
- clicking on  $f.\boxplus$ : when the user click on  $f.\boxplus$  the intention *ctx.Int* of the model is changed. Specifically, the measuring function is defined as:  $mE' = mE + f$ , where *f* can again denote a facet or a property path. If *f* corresponds to a value, then a restriction on the measuring function is applied.

In both cases, the extension, i.e.,  $ctx.Ext$ , as well as the transitions remains the same, since these buttons do not affect neither the displayed objects on the right frame nor the available transition markers.

In addition, as we mentioned earlier, for supporting HAVING queries, the result set should be loaded as a new dataset to the system. That will also enable users to define analytical queries of unlimited nesting depth.

## 6 The interaction model formally and the related algorithms

In Sect. 6.1, we introduce the notations that we shall use for describing (in Sect. 6.2) the desired state space of the proposed interaction declaratively (the procedural specification is given later in Sect. 7), while in Sect. 6.3 we describe how a result set can be reloaded as a new dataset.

### 6.1 Notations

**RDF.** Let  $K$  be a set of RDF triples and let  $\mathcal{C}(K)$  be its closure (i.e., the set containing also the inferred triples). We shall denote with  $C$  the set of classes, with  $Pr$  the set of properties, with  $\leq_{cl}$  the `rdfs:subClassOf` relation between classes, and with  $\leq_{pr}$  the `rdfs:subPropertyOf` relation between properties. We also define the instances of a class  $c \in C$  as  $inst(c) = \{o \mid (o, rdfs:type, c) \in \mathcal{C}(K)\}$  and the instances of a property  $p \in Pr$  as  $inst(p) = \{(o, p, o') \mid (o, p, o') \in \mathcal{C}(K)\}$ .

For defining formally the *transitions*, we provide some auxiliary definitions, too. We shall denote with  $p^{-1}$  the *inverse* direction of a property  $p$ , e.g., if  $(d, p, r) \in Pr$ , then  $p^{-1} = (r, inv(p), d)$ , and with  $Pr^{-1}$  the inverse properties of all properties in  $Pr$ .

Below, we introduce notations for *restricting* the set of resources  $E$ ;  $p$  is a property in  $Pr$  or  $Pr^{-1}$ ,  $v$  is a resource or literal,  $vset$  is a set of resources or literals, and  $c$  is a class.

$$\begin{aligned} Restrict(E, p : v) &= \{e \in E \mid (e, p, v) \in inst(p)\} \\ Restrict(E, p : vset) &= \{e \in E \mid \exists v' \in vset \text{ and } (e, p, v') \in inst(p)\} \\ Restrict(E, c) &= \{e \in E \mid e \in inst(c)\} \end{aligned}$$

We also provide a notation for *joining* values, i.e., for computing values which are linked with the elements of  $E$ :

$$Joins(E, p) = \{v \mid \exists e \in E \text{ and } (e, p, v) \in inst(p)\}$$

### 6.2 Defining the state space of the interaction

Here, we describe the transitions between the states followed by examples, for understanding the sought interaction.

**Interaction States.** If  $s$  denotes a *state*, then we shall use  $s.Ext$  to denote its *extension*. Let  $s_0$  denote an artificial (or default) *initial state*. We can assume that  $s_0.Ext = URI \cup LIT$ , i.e., the extension of the initial state contains (1) every URI and literal of the dataset, i.e., all individuals,<sup>6</sup> or a subset of the dataset, e.g., the result of a keyword query [15], or of a natural language query [56].

<sup>6</sup> i.e., the results of the SPARQL query “select ?x where { ?x rdfs:type owl:NamedIndividual. }.”

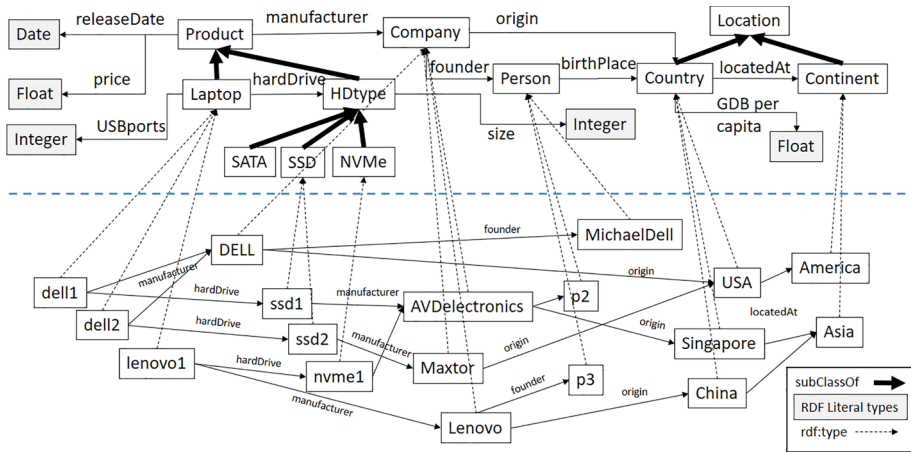


Fig. 7 Data of our running example

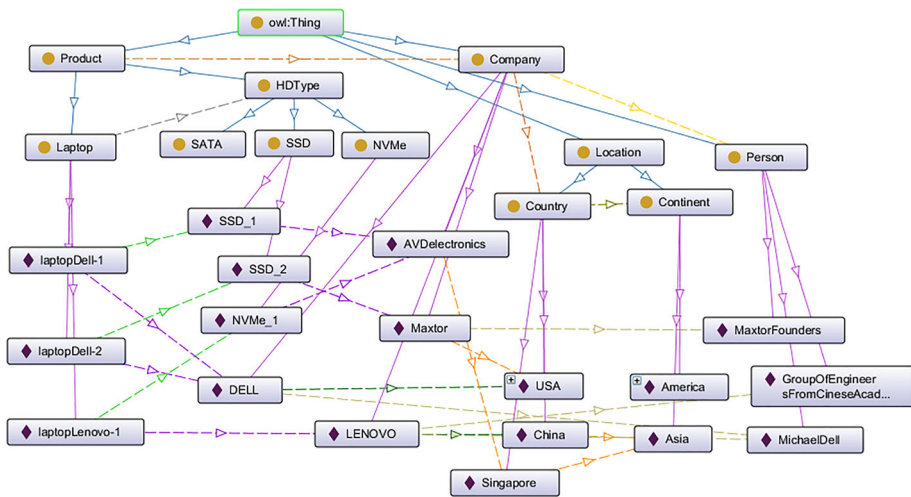


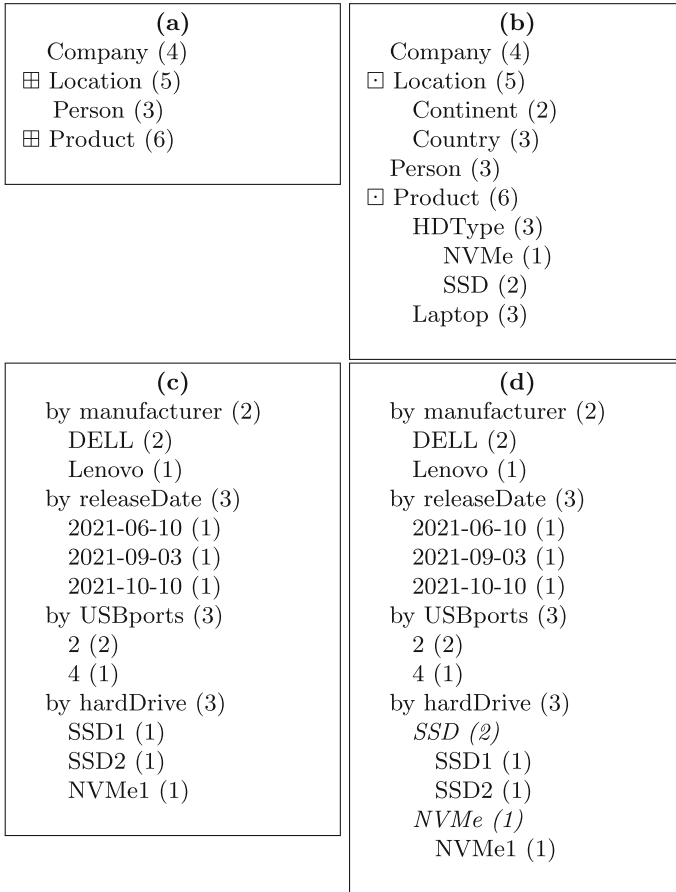
Fig. 8 Data of our running example (as visualized by Protege)

### 6.2.1 Initial class-based transitions

Below we shall refer to examples assuming the schema of Fig. 1; in particular, we consider a few instances, specifically the ones illustrated in Figs. 7 and 8; the latter is created by Protege; however, in both cases the diagrams show only the more important information.

Initially, the top-level or maximal classes  $maximal_{\leq_{cl}}(C)$  are presented, see Fig. 9a.

Each of these classes corresponds to a “class-based transition marker” and leads to a state with extension  $inst(c)$ . These classes can be expanded and unfold their subclasses (see Fig. 9 (b)) as well as their top-level or maximal properties  $maximal_{\leq_{pr}}(Pr)$  (see Fig. 9c). Each subclass  $c \in subclasses_{\leq_{cl}}(C)$  corresponds to a “class-based transition marker.” Each such transition yields again a state with extension  $inst(c)$ . If the number of the subclasses is high, then they are hierarchically organized based on the subClass relationships among these



**Fig. 9** **a** Class-based transition markers, **b** class-based transition markers expanded, **c** property-based transition markers, **d** property-based transition markers and grouping of values

classes. Such a layout illustrates the structure of the *reflexive and transitive reduction*<sup>7</sup> of the restriction of  $\leq_{cl}$  on the applicable transition markers  $C$  (i.e., on  $R^{refl,trans}(\leq_{cl} | C)$ ).

### 6.2.2 Property-based transitions

Having expanded the top-level classes, their maximal properties  $maximal_{\leq_{pr}}(Pr)$  unfold (Fig. 9c). Each applicable value of these properties corresponds to a “property-based transition marker.” Specifically, if  $E$  is the set of current objects, the property-based transitions by  $p$  are the set  $Joins(E, p)$ . By clicking on a value  $v$  in  $Joins(E, p)$ , we transit to a state with extension  $Restrict(E, p : v)$ .

If the number of the subproperties is high, then they are hierarchically organized based on the `subproperty` relationships among these properties.

<sup>7</sup> The reflexive and transitive reduction in a binary relation  $R$  is the smallest relation  $R'$  such as both  $R$  and  $R'$  have the same reflexive and transitive closure.



(a)	(b)
by manufacturer (2)	by manufacturer (3) ▷ by origin (2)
DELL (2)	DELL (2)      US (1)
Lenovo (1)	Lenovo (1)      China (1)
by releaseDate (3)	by releaseDate (3)
2021-06-10 (1)	2021-06-10 (1)
2021-09-03 (1)	2021-09-03 (1)
2021-10-10 (1)	2021-10-10 (1)
by USBports (3)	by USBports (3)
2 (2)	2 (2)
4 (1)	4 (1)
by hardDrive (3)	by hardDrive (3) ▷ by manufacturer (2) ▷ by origin (2)
SSD1 (1)	SSD1 (1)      Maxtor (2)      Singapore (1)
SSD2 (1)	SSD2 (1)      AVDElectronics (1)      US (1)
NVMe1 (1)	NVMe1 (1)

**Fig. 10** **a** Property-based transition markers, **b** property path-based transitions markers

**Transitions for Path Expansion.** Let  $p_1, \dots, p_k$  be a sequence of properties. We shall call this sequence *successive*, if  $Joins(Joins(\dots(Joins(s.Ext, p_1), p_2) \dots p_k) \neq \emptyset$ , i.e., if such a sequence does not produce empty results. Let  $M_1, \dots, M_k$  denote the corresponding sets of transition markers at each point of the path. If we assume that  $M_0 = s.Ext$ , then the transition markers for each  $i$ , where  $1 \leq i \leq k$ , are defined as:  $M_i = Joins(M_{i-1}, p_i)$  (Fig. 10).

Now suppose that the user selects a value  $v_k$  from the transition marker  $M_k$  (i.e., one value from the end of the path). Such an action will restrict the set of transitions markers in the following order  $M_k, \dots, M_1$ , and finally, it will restrict the extension of  $s$ . Let  $M'_k, \dots, M'_1$  be these restricted sets of transitions markers. They are defined as  $M'_k = \{v_k\}$  (this is the value that the user selected from the end of the path), while for  $1 \leq i < k$  they are defined as:

$$M'_i = Restrict(M_i, p_{i+1} : M'_{i+1}) \tag{1}$$

The extension of the new state  $s'$  is defined as  $s'.e = Restrict(s.Ext, p_1 : M'_1)$ . Equivalently, we can consider that  $M'_0$  corresponds to  $s'.e$ , and in that case Eq. 1 holds also for  $i = 0$ .

### 6.3 Loading AF as a new dataset

The results of an analytic query can be loaded as a new derived (RDF) dataset that the user can further explore and restrict. Assume that the answer of the current analytic query is a table with attributes  $A_1, \dots, A_k$ , comprising a set of tuples  $T = \{t_1, \dots, t_n\}$ . We assign to each tuple  $(t_{i1}, \dots, t_{nk})$  a distinct identifier, say  $t_i$ , and we produce the following  $k$  RDF triples:  $(t_i, A_j, t_{ij})$  for each  $j = 1 \dots k$ . These  $n * k$  triples are loaded to the system, and they can be explored as if they were an ordinary RDF dataset. Any restriction expressed over this model corresponds to HAVING clauses over the original data.

## 7 The algorithm that implements the state space

Here, we provide the exact algorithm for building the GUI of the proposed model that will be in compliance with the state space described in Sect. 6.2. Below we describe each step of the algorithm.

## 7.1 Starting points

The function **Startup** shows that the interaction can start in two ways: (1) from scratch, or (2) by exploring a set, let's call it *Results* provided by an external access method (e.g., a keyword search query). In both cases, the function **ComputeNewState** is called. The responsibility of this function is to compute and display the facets of the left frame and the objects of the right frame.

## 7.2 Computing the objects in the right frame

The computation of the objects in the right frame is described in the **Part A** of the algorithm Alg. 1. As we can see the parameter *Filt* can be equal to "CLASS *c*," "PVALUE *p*:*v*," "PVALUE *p*:*vset*," or empty ( $\epsilon$ ). If empty, the current set *E* is set to be all objects of the KG. If nonempty, it restricts the current set of resources *E* according to the notations given in Sect. 6.1.

---

### Algorithm 1 Computing Active Facets, Zoom points (Filters) and Analytics

---

```

1: function Startup                                     ▷ Two ways to start the exploration process
2:   ComputeNewState( $\emptyset$ ,  $\epsilon$ ,  $s_0$ )                   ▷ Initial call if we want to explore the entire KB
3:   ComputeNewState(Results,  $\epsilon$ ,  $s_0$ )                 ▷ Initial call if we want to explore a particular set of objects
   (Results) coming from an external access method
4: end function

5: function ComputeNewState(E:set of objects, Filt: Restriction spec, s: current state)
  ▷ Part A: Computation of the objects for the right frame
6:   if  $E = \emptyset$  then                               ▷ meaning that we have to explore the entire KB
7:      $E \leftarrow Obj$                                    ▷  $E$  is set to the set of all objects
8:   else if  $Filt \neq \epsilon$  then                       ▷ if we have to compute a restriction of the current state
9:     switch (Filt):                                   ▷ Computation of the restricted  $E$  for the right frame
10:      case "CLASS  $c$ ":                                $E \leftarrow Restrict(s.Ext, c)$ 
11:      case "PVALUE  $p : v$ ":                            $E \leftarrow Restrict(s.Ext, p : v)$ 
12:      case "PVALUE  $p : vset$ ":  $E \leftarrow Restrict(s.Ext, p : vset)$ 
13:    endSwitch
14:   end if
15:   Show  $E$                                              ▷ Display the objects in  $E$  in the right frame

  ▷ Part B: Computation of the facets for the left-frame
  ▷ Part B.1: Computation of class-based restrictions
16:    $FacetsClasses \leftarrow TM_{cl}(E)$                  ▷ The applicable class-based transition markers
17:   for each  $c \in FacetsClasses$  do                   ▷ Creation of the nodes for the transition markers (with names,
   counts, and onClick)
18:     Node node  $\leftarrow$  new Node();
19:     node.name  $\leftarrow$   $c.name$ ;                       ▷ The name that will be displayed
20:     node.count  $\leftarrow$   $| Restrict(s.Ext, c) |$        ▷ The count that will accompany the name
21:     node.onClick  $\leftarrow$  ComputeNewState( $E$ , "CLASS  $c$ ",  $s$ ) ▷ the onClick behavior
22:   end for
23:   Part B.2 see Alg. 2
24: end function

```

---

### 7.3 Computing the facets corresponding to classes

The part of the algorithm for computing the facets of classes is **Part B.1** of Alg. 1. Being at a state  $s$  with extension  $E$ , the classes that can be used as class-based transition markers, denoted by  $TM_{cl}(E)$ , are those that the entities in  $E$  belong to, and they are defined as:

$$TM_{cl}(E) = \{c \in C \mid Restrict(E, c) \neq \emptyset\} \tag{2}$$

If the user clicks on a class-based transition marker, i.e.,  $c \in TM_{cl}(E)$ , then the extension of the targeting state  $s'$  is defined as  $s'.e = Restrict(E, c)$ .

For each such transition marker, a *node* is created. Each node has a *name*, i.e., the name of the corresponding subclass, a *count* information, i.e., the number of entities that belong to this subclass, and an *on-click behavior*. Clicking on the node, the function “ComputeNewState” is called. In that case, where the objects are restricted to a class, the filtering condition that is passed to the call of this function is “Class C”, i.e.,  $ComputeNewState(E, \text{“CLASS } c\text{”}, s)$ .

### 7.4 Computing the facets that correspond to properties

The part of the algorithm for computing the facets that correspond to properties is **Part B.2** of Alg. 2.

---

#### Algorithm 2 Computing Active Facets, Zoom points (Filters), and Analytics

---

```

1: function ComputeNewState( $E$ :set of objects, Filter: Restriction spec,  $s$ : current state)
    ▷ Part B.2: Computation of property-based restrictions
2:    $FacetsPropsForw \leftarrow \{p \in Pr \mid Joins(s.Ext, p) \neq \emptyset\}$            ▷ forward properties
3:    $FacetsPropsBack \leftarrow \{p \in Pr^{-1} \mid Joins(s.Ext, p) \neq \emptyset\}$    ▷ backwards props
4:   for each  $p \in FacetsPropsForw$  do
5:     Node  $hnode \leftarrow$  new HeadingNode( $p.name$ )           ▷ Separator and name of  $p$ 
6:     for each  $v \in Joins(s.Ext, p)$  do                       ▷ TMs related to  $p$ 
7:       Node  $node \leftarrow$  new Node();
8:        $node.name \leftarrow v$ ;                               ▷ The name that will be displayed
9:        $node.count \leftarrow \mid Restrict(s.Ext, p : v) \mid$    ▷ The accompanying count
10:       $node.onClick \leftarrow$  ComputeNewState( $E, \text{“PVALUE } p : v\text{”}, s)$ 
11:    end for
12:    Optional: group all values based on their classes, i.e., with respect the following classes:  $\{c \mid Joins(s.Ext, p) \cap inst(c) \neq \emptyset\}$ 
13:     $hnode.addButton(\text{Ⓜ}, onClick=gE += p)$                  ▷ For group by
14:     $hnode.addButton(\text{Ⓢ}, onClick=mE += p)$                ▷ For measuring op
15:    ▷ For Entity Type Switch:
16:     $hnode.addButton(\text{Entity Type Switch},$            ▷ For entity type switch
17:       $onClick \leftarrow$  ComputeNewState( $Joins(s.Ext, p), \epsilon, s)$ )
18:    ▷ For Path Expansion:
19:     $hnode.addButton(\text{“▷”},$                              ▷ For path expansion
20:       $onClick \leftarrow$  ComputeNewState( $s.Ext,$ 
21:         $\text{“PVALUE } p\text{”} + \text{ExpandAndRestrictRecursive}(Joins(s.Ext, p)), s)$ 
22:    end for
23:    ...           ▷ Analogously for  $FacetsPropsBack$ 
24: end function

```

---

In brief, being at a state  $s$ , with extension  $E$ , the properties that can be used in the expansion of a property  $p$  are those that connect to that property as well as to the current entities  $E$  of

focus, and they are defined as:

$$P = \{p \in Pr \mid Joins(E, P) \neq \emptyset\} \quad (3)$$

For each such property  $p$ , a heading node is created. For each property value of  $p$  that is joinable, a node is created with the appropriate name, count, and on-click behavior. Moreover, two buttons for analytics, i.e.,  $\square$  and  $\boxplus$  are created. Finally, an additional expansion button, i.e., “▷”, is added enabling the user to further expanding the property path (described next in Sect. 7.4.1).

Clicking on a node corresponding to a property value, the function “ComputeNewState” is called. In that case, where the objects are restricted over a property, the filtering condition that is passed to the function is “PVALUE p:v”, i.e.,  $ComputeNewState(E, \text{“PVALUE p:v”}, s)$ .

As regards the part of the algorithm about *Entity Type Switch* (line 15), this function enables the user to set as focus (new state) the transition markers of a property. This can be useful for navigating through different types of entities connected by the chosen property. For instance, while the user is filtering products based on their manufacturer and other filters, he can decide to switch the type, and start exploring (and filtering out) these manufacturers. For achieving this, it is enough to set a extension of the new state and joins of the property (line 17).

### 7.4.1 Computing the facets corresponding to path expansion

Notice in Alg. 2 the line about *path expansion* (line 18). On clicking on the element “▷”, the algorithm for computing the facets that correspond to path expansion is called, shown in Alg. 3. By clicking on “▷”, the function “ExpandAndRestrictRecursive(M)” is called again and the process is repeated (as described in Sect. 6.2.2).

---

#### Algorithm 3 Function for Path Expansion

---

▷ Carries out the expansion over a set of URIs  $M$  and returns a set of values (to be used for filtering by the caller).

```

1: function EXPANDANDRESTRICTRECURSIVE( $M$ :Uris):ValuesSet
2:    $P \leftarrow \{p \in Pr \mid Joins(M, p) \neq \emptyset\}$                                 ▷ applicable properties
3:   for each  $p \in P$  do
4:     Node hnode = new HeadingNode(p.name)                                       ▷ Separator and name of  $p$ 
5:     for each  $v \in Joins(M, p)$  do                                             ▷ TMs related to  $p$ 
6:       Node node  $\leftarrow$  new Node();
7:       node.name  $\leftarrow v$ ;                                                 ▷ The name that will be displayed
8:       node.count  $\leftarrow | Restrict(M, p : v) |$                                ▷ The accompanying count
9:       node.onClick  $\leftarrow$  return  $Restrict(M, p : v)$                          ▷ on click it returns a set
10:    end for
11:    hnode.addButton(“▷”,                                                       ▷ recursive call for further expansion
12:      onClick  $\leftarrow$  return  $Restrict(M, p, ExpandAndRestrictRecursive(Joins(M, p)))$ 
13:    hnode.addButton( $\square$ , onClick= $gE+ = p$ )                                     ▷ For group by
14:    hnode.addButton( $\boxplus$ , onClick= $mE+ = p$ )                                     ▷ For measuring op
15:  end for
16: end function

```

---

**Table 2** SPARQL expression of the model’s notations, assuming that the extension of the current state (either  $E$  or  $s.Ext$ ) is stored in temporary class  $\text{temp}$

Id	Notation	Expression in SPARQL
(i)	$Restrict(E, p : vset)$ , where $vset = \{v_1, \dots, v_k\}$	<pre>select ?x where {   ?x rdf:type temp; p ?V.   Filter (V=\$v_1   ...   ?V=v_k)}</pre>
(ii)	-//-	<pre>select ?x where {   VALUES ?x { e1 ... en}.   ?x p ?V.   Filter (?V=v_1   ...   ?V=v_k)}</pre>
(iii)	$Restrict(E, c)$	<pre>select ?x where {   ?x rdf:type temp; rdf:type c.}</pre>
(iv)	$Joins(E, p)$ , where $E = \{e_1, \dots, e_k\}$	<pre>select Distinct ?v where { ?x p ?v.   Filter (?x = e_1   ...   ?x = e_k)}</pre>
(v)	$TM_{cl}(s.Ext)$ and counts	<pre>select Distinct ?c count(*) where{   ?x rdf:type ?c; rdf:type temp.} group by ?c</pre>
(vi)	$Props(s)$	<pre>select Distinct ?p where{ {?x rdf:type temp; ?p ?v.} UNION {?m rdf:type temp. ?n ?p ?m. }}</pre>
(vii)	$Joins(s.Ext, p)$ and counts	<pre>select Distinct ?v count(*) where{ ?x rdf:type temp; p ?v.} groupby ?v</pre>

**Table 3** For SPARQL-only evaluation approach

Notation	Expression in SPARQL
$E \leftarrow Restrict(s.Ext, c)$	$s.q \leftarrow s.q + "?x1 rdf:type c"$
$E \leftarrow Restrict(s.Ext, p : v)$	$s.q \leftarrow s.q + "?x1 p v"$
$E \leftarrow Restrict(s.Ext, p : vset)$	$s.q \leftarrow s.q + "?x1 p ?V. Filter (?V=v1   ...   ?V=vk)"$

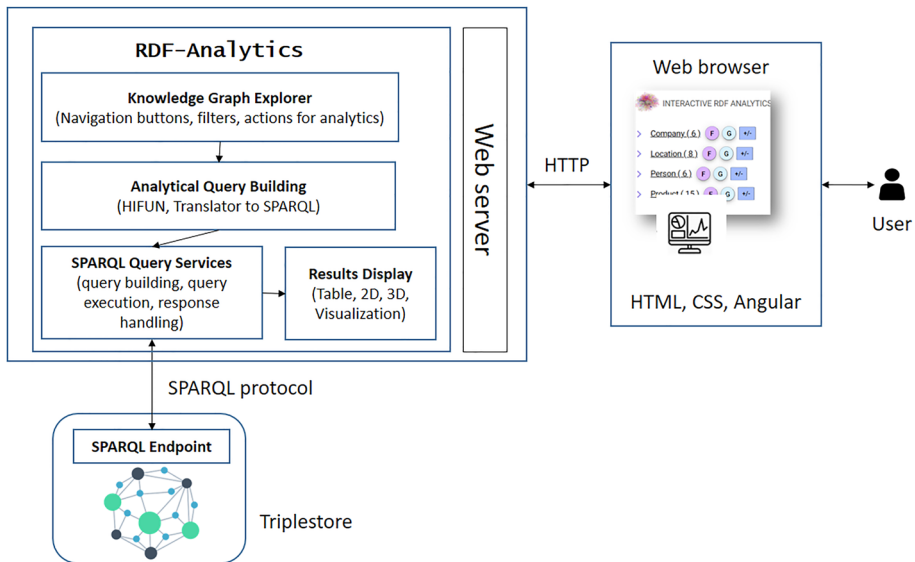
### 8 Expressing and computing the intentions of the states

Here, we explain how the intentions of the proposed model are expressed in a specific query language that of RDF data, i.e., SPARQL.

Table 2 (adapted from [21]) interprets the notations specified for this model and shows how the intentions are expressed in SPARQL. We assume that all the inferred triples (deduced from `subClassOf` and `subPropertyOf` relations) are available (materialized) in the storage level.

As it is shown, the extension of the current state (i.e.,  $ctx.Ext$ ) can be computed either (1) *extensionally* or (2) *intentionally*. “Extensional” means that the current state is stored in a temporary class  $\text{temp}$ , i.e., the RDF triples ( $e, \text{rdf:type}, \text{temp}$ ) for each  $e \in ctx.Ext$  have been added to the triplestore. On the other hand, “intentionally” means that instead of storing the current state in a temporary class, the desired triples are obtained by querying the triplestore. The way queries are formulated is given in Table 3.

The approach to be selected (extensional or intentional) depends on the size of the dataset and the server’s main memory. In particular, if the dataset is quite big and cannot not fit in



**Fig. 11** Architecture of the system RDF-ANALYTICS

the memory, then the intentional approach is preferred. In our implementation, described in the next section, we adopt the intentional approach.

## 9 Implementation

We have implemented the proposed model as a web application, called **RDF-ANALYTICS**. The server-side uses the triplestore Virtuoso<sup>8</sup> that offers persistent storage and the SPARQL endpoint for executing the queries. The front-end side of the system is implemented in Angular.<sup>9</sup> The states are computed using the intentional approach, i.e., the extension of the new state is computed using a SPARQL query without based on the extension of the previous state. Figure 11 shows the main components of the implementation and the responsibilities of each component.

Figure 12 shows a screenshot of the left frame of the GUI representing the data of our running example. Here, the user has expressed the query “Average price of laptops that have been manufactured by US companies and they have from 2 to 4 USB ports, group by manufacturer”. The results of this query are initially displayed in tabular form as shown in Fig. 13 (a). At this point, the user can (1) change the attributes of analysis (by clicking on the “BACK” button), (2) export the results as a.csv file (by clicking on the “EXPORT AS EXCEL” button), (3) visualize the results (by clicking on the “VISUALIZE” button) as shown in Fig. 13b, or (iv) further restrict the final results (by clicking on the “EXPLORE WITH FS” button) as shown in Fig. 13c.

**Common Extensions.** The system could be enriched with auxiliary functionalities that are useful in case the ontology is too big, i.e., it contains numerous classes and schema properties, and the values that are used as property values are too many. Such extensions

<sup>8</sup> <http://docs.openlinksw.com/virtuoso/>.

<sup>9</sup> <https://angular.io/>.

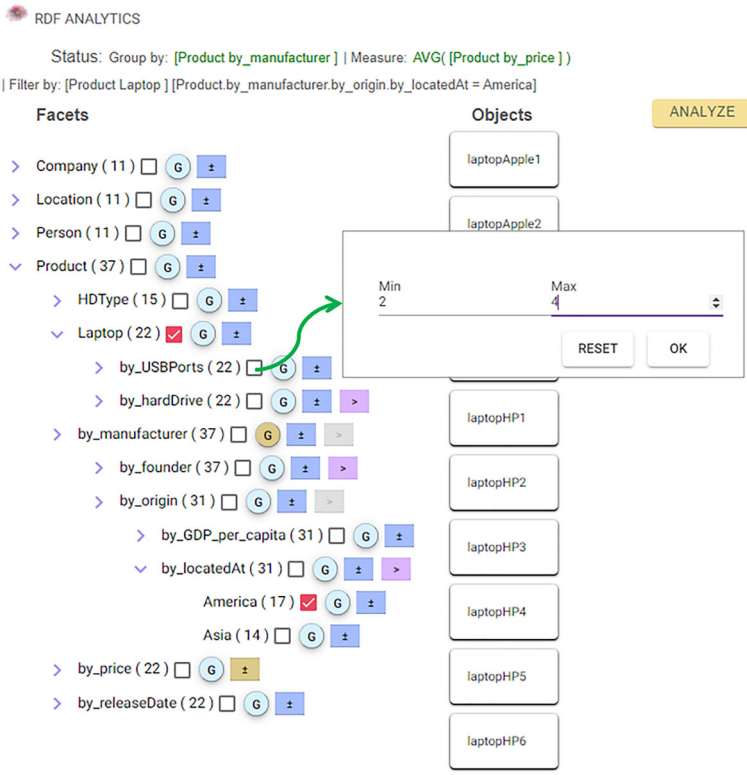


Fig. 12 GUI of the system RDF-ANALYTICS for formulating the query “Average price of laptops that have 2–4 USB ports and have been manufactured by US company, group by manufacturer”

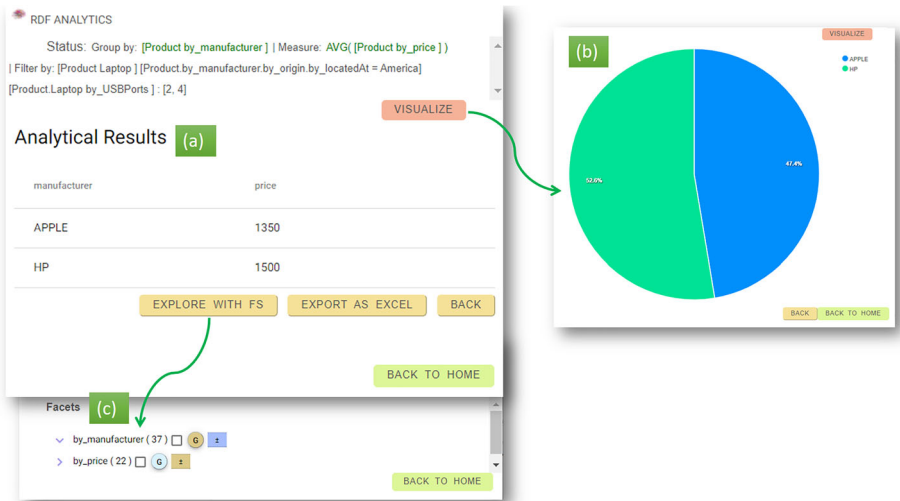


Fig. 13 Tabular and graphical visualization of the results—loading of the results as a new dataset

include: *search boxes* on each individual facet, methods for *facet/value ranking* (as described in Sect. 6.3.2 of the survey [21] and more recent ones like [35, 57]), methods for predicting useful facets [58], as well as with *similarity-based browsing* functions (as in [14]). Currently, we are investigating the connection of the system with a 3D visualization (based on [59]) for enabling a more interactive exploration of the results.

## 9.1 Efficiency

**Complexity and Optimizations.** The implementation does not have any additional cost except for the evaluation of the respective SPARQL queries. As described in the previous section (Sect. 8), there is an extensional and an intentional approach. The intentional is beneficial for big datasets. Below we discuss the time complexity of the algorithms, for the intentional implementation, i.e., for computing the required queries. In particular, part A of the main algorithm (that computes the extension of the new state) evaluates a SPARQL query that performs a simple restriction. Part B.1 (that computes the class-based transition markers and their count information), evaluates a SPARQL query (like the one shown in row (5) of Table 2) that finds and counts the instances of a particular class. Part B.2 (that computes the property-based transition markers and their count information), evaluates a SPARQL query that (1) fetches and (2) counts the range values of the properties that connect to the current resources (as shown in rows (4) and (7) of Table 2, respectively), so essentially it relies on the evaluation a couple of queries. The algorithm for path expansion (Alg. 3) evaluates the queries of Part B.2 of the main algorithm for each expansion step.

The efficiency of queries' execution depends on (1) the storing, indexing, and query processing techniques of the triplestore (as discussed in [55]) and (2) the size of the collection. Query optimization is beyond the scope of this work, i.e., our focus is how to facilitate the formulation of the query. We should mention that there are works that focus on optimizations for the evaluation of SPARQL analytic queries. Apart from the corresponding parts of the surveys [37, 55], we could mention that [60] focuses on queries that include several chain and star patterns.

[61] focuses on the optimization of SPARQL queries with aggregate operators; in particular, it implements pattern matching queries with the help of two index structures, a VS\*-tree (which is a specialized B+-tree) and a trie-based T-index; [62] focuses on the optimization of aggregate queries over federations of SPARQL endpoints by materializing the intermediate results of the queries; in particular, it proposes a cost model for estimating result sizes for components, such as triple patterns, joins, grouping, and aggregation. These components are combined with the processing strategies into an approach called the Cost-based Optimizer for distributed aggregate queries. Finally, [63] focuses on the selection and materialization of aggregate RDF views for reasons of efficiency and provides the needed rewriting method for user queries.

**Experimental Evaluation.** Below, we report a few indicative execution times of such queries for datasets of various sizes. Based on the schema of the running example, we produced synthetic datasets of different sizes from 100 triples to 1 million triples. Then, we tested the following list of queries regarding the time needed for executing them as well as displaying the final results to the user. The set of queries ranges from simple to more complex containing groups, paths, filters, and combinations of them.

- Q1: Average price of laptops group by manufacturer (simple group)
- Q2: Average price of laptops group by manufacturer and release date (combination of groups)



- Q3: Average price of laptops that have at least 2 USB ports group by manufacturer (filter)
- Q4: Average price of laptops group by the birthplace of founders (path)
- Q5: Average price of laptops released after 1/1/2021, and have at least 2 USB ports, group by the origin of manufacturer (complex query containing path, filters, group)
- Q6: Average price of laptops released after 1/1/2021, have at least 2 USB ports, and have gdb of origin [2000, 15,000], group by the origin of manufacturer and the birthplace of the founders (complex query containing paths, filters, groups)

We carried out experiments on a laptop equipped with an Intel(R) Core(TM) i5-7200U CPU, operating at a base clock speed of 2.50GHz and a turbo boost frequency of 2.70GHz.

The system was configured with 8.00 GB of RAM and used a 64-bit operating system, specifically Windows 10, version 22H2, on an x64-based processor architecture. As it is shown in Table 4, the response time depends mainly on the evaluation of the query, since the time required for displaying the results to the user is negligible.

We also notice that the functions and the aggregate operations that the query has do not affect its execution time. Overall, as we can see, we support real-time interaction. In general, the evaluation time of the query depends on the storage, indexing, and query processing techniques that the SPARQL endpoint uses (in this case Virtuoso).



## 10 The expressive power of the model

We aim to cover the more basic information needs in a familiar interaction style and not to propose an interaction model with extreme expressive power that would be complex to use. In such cases, the users could directly use SPARQL. Just like Faceted Search systems which offer an intuitive and widely successful method for incrementally formulating mainly conjunctive queries (or conjunctions over facet-restricted disjunctions), our goal is to support the main needs for analytic queries over RDF.

Below, we describe the expressive power of the proposed model with respect to (a) the kind of HIFUN expressions that can be formulated by this model (in Sect. 10.1) and (b) the OLAP operations it supports (in Sect. 10.2).


### 10.1 Expressible HIFUN queries



Here, we discuss about the kind HIFUN queries that can be expressed with the proposed model.

- *Analysis context*: it is the set  $D$  of the objects of focus, i.e., *ctx.Ext*. The proposed model lets the users define this set through the facets it offers, since they restrict the information space and specify the focus.
- *gE* is the grouping expression: refers to the facet(s) by which the analytical results are grouped. The facet(s) may belong to different categories, e.g., classes (pairing operator in HIFUN) and a facet can correspond to a path (composition operator in HIFUN). Since a  button is laid next to each facet name of any level, the system supports both pairing and composition expressions for the grouping function.
- *mE* is the measuring expression: refers to the facet(s) on which the aggregate operation will be applied. Again, the facet(s) may belong to different categories (pairing operator in HIFUN) and a facet can correspond to a path (composition operator in HIFUN). Since a  button is laid next to each facet name of any level, the system supports both pairing and composition expressions for the measuring function.

**Table 4** Efficiency

Dataset size (in triples)	Query evaluation (in ms)						Presentation of results (in ms)					
	q1	q2	q3	q4	q5	q6	q1	q2	q3	q4	q5	q6
$10^2$	345	425	388	364	624	418	24	25	29	25	45	26
$10^3$	331	415	452	409	521	618	23	32	56	37	30	49
$10^4$	439	409	440	381	454	488	26	34	33	44	32	30
$10^6$	2605	3258	2276	2544	1937	1569	58	62	59	111	61	47

- *opE* the operation expression: it is the aggregation function applied to the grouped values. Since a  button laid next to each facet name offers a menu with the basic aggregation functions of an analytical query, the system supports this function, too.

**Attribute restrictions.** Recall that the grouping and the measuring function may contain a set of restrictions. Any restriction is also supported by the proposed model, since the  and  buttons lay next to each value of every facet.

**Results restrictions.** Apart from restricting the attributes of a HIFUN query, the user can also restrict the results of it. The proposed model supports this functionality by loading the results of the analytical query as a new dataset on the FS system. In that case, the user is able to further restrict them by specifying the range of the values of the desired facets.

**Nesting.** A HIFUN query may express a query in terms of one or more other queries, i.e., nested queries. Since the proposed system lets the users load the current analytical results as a new dataset to the system and create new queries, it supports nested queries, too.


## 10.2 OLAP operators supported

*Online Analytical Processing systems (OLAP).*

Online Analytical Processing (OLAP), introduced in the early '90s [64], is used for the analysis of transaction data.

In order to apply OLAP, data should be organized in a multi-dimensional (MD) structure, known as *Data Cube*. A Data Cube consists of (1) facts which are the subjects of the analysis and quantified by measures and (2) hierarchically organized dimensions allowing for measure aggregation. The operations that can be applied over it are: *roll-up* (performs aggregation by climbing up a concept hierarchy for a dimension or by dimension reduction), *drill-down* (is the reverse operation of roll-up and performs aggregation by stepping down a concept hierarchy for a dimension or by introducing a new dimension), *slice* (selects one particular dimension from a given cube and provides a new sub-cube), *dice* (selects two or more dimensions from a given cube and provides a new sub-cube), and *pivot* (provides an alternative presentation of data).

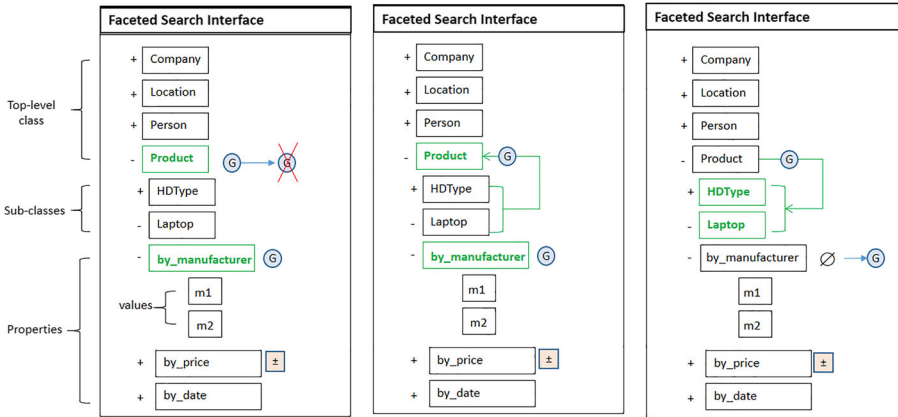
The comparison of the exploration capabilities of plain Faceted Search with OLAP has been described long ago, e.g., in Sect. 3 of [20]. The model presented in the current paper is about graph data, and we have seen that it can be applied directly over the graph, i.e., without requiring to define first a data cube, as it is discussed in Sect. 3.2. The additions that we propose (to FS for enabling analytic queries) essentially cover the needs for OLAP over graph data. Assume an analytic query expressed with `RDF-ANALYTICS`. The user by applying restriction and group by actions can achieve the desired result, i.e., the desired level of granularity and the desired sub-cube. The correspondence is illustrated in Fig. 14. In particular, *traversing up* the hierarchy of a facet, or *removing adding a GroupBy action*, corresponds to a *roll-up* operation, *traversing down* the hierarchy of a facet, or *adding a new GroupBy action*, corresponds to a *drill-down* operation, *picking one* value for a facet corresponds to *slice*, *picking two* or more values from multiple facets corresponds to *dice*, and *moving to* a facet which is directly or indirectly connected to the facet of focus corresponds to *pivot*.

To make this more clear below, we provide one particular example of roll-up and drill-down. Suppose a user who has expressed the query “Average prices for products, grouped by manufacturer”, as shown in Fig. 15(left). If the user wants to drill down and inspect the average prices also based on the product type, he can press the  button on the class Product. That

**Roll-up:** aggregating data by removing a dimension

**Roll-up:** aggregating data by climbing up a concept hierarchy

**Drill-down:** aggregating data by descending down a concept hierarchy, or adding a dimension



**Slice:** selecting or filtering a specific value along a single dimension

**Dice:** selecting or filtering specific values along multiple dimensions

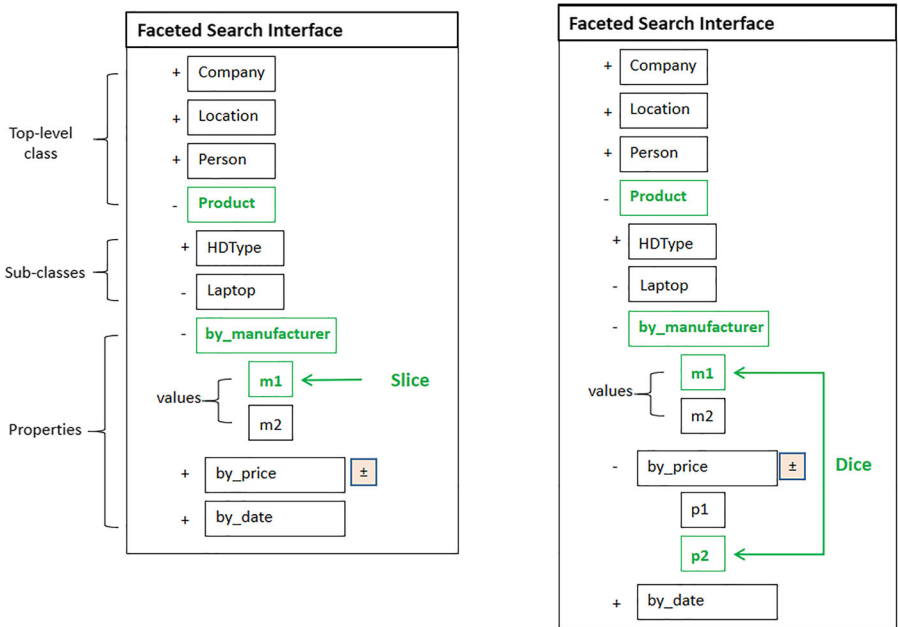


Fig. 14 Correspondence with OPAP operations

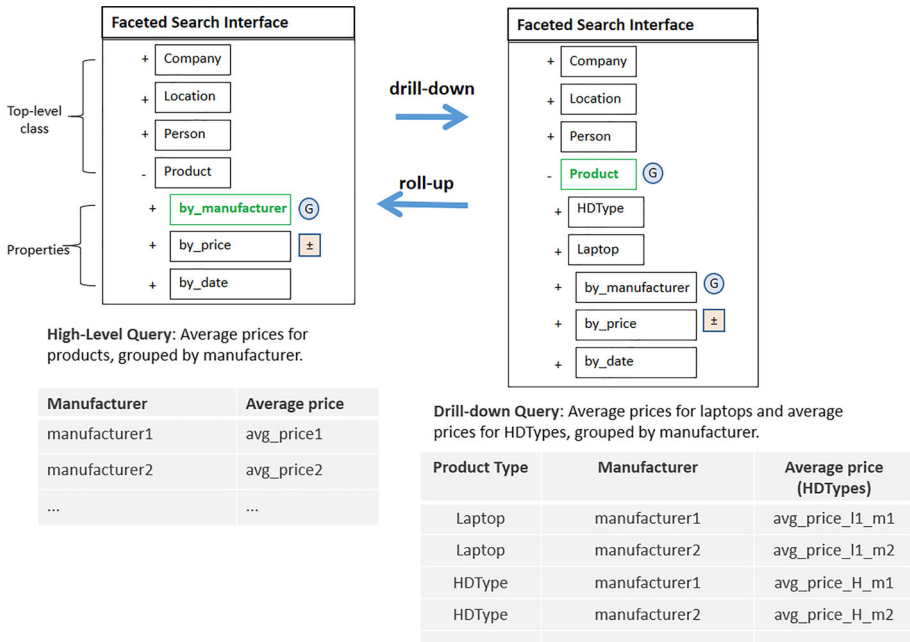


Fig. 15 An example of roll-up and drill-down

would correspond to the query “average prices of product types grouped by manufacturer”. This is shown in Fig. 15(right). The inverse direction would be roll-up.

## 11 Evaluation

Section 11.1 discusses the results of an evaluation of our approach with users, and Sect. 11.2 conveys an implementation of the model that proves its feasibility and the completeness of the introduced algorithms.

### 11.1 Task-based evaluation with users

We performed a task-based evaluation with users. The objective was to investigate if they can formulate easily analytic queries, especially complex queries containing various value restrictions and path expressions. Twenty (20) users participated to the evaluation. The number was sufficient for our purposes since, according to [65], 20 evaluators are enough for getting more than 95% of the usability problems of a user interface. The participants had varying educational levels (Computer Science Student (25%), Computer Science related (60%), Other (15%)), level of experience (experts (55%), medium knowledge of RDF and SPARQL (30%), novice (15%)), age groups (twenties (25%), thirties (40%), forties (20%), fifties (10%), sixties (5%)), and sex (male (80%), female (20%)). We did not train them; we just provided them with a concise assisting page that explains the functionality of the buttons laid next to each facet.<sup>10</sup>

<sup>10</sup> The deployment of the system that was used is accessible at <https://demos.isl.ics.forth.gr/rdf-analytics>.

We defined 10 tasks for the evaluation; below we list them along with the success rates of the users.

- Q1. Count the laptops grouped by manufacturer.: Success (85%), Partial success (0%), Fail (15%)
- Q2. Count the number of laptops grouped by manufacturer that were released after 1/1/2022.: Success (85%), Partial success (0%), Fail (15%)
- Q3. Count the number of laptops grouped by manufacturer that were released after 1/1/2022 and have at least 2 USB ports.: Success (85%), Partial success (0%), Fail (15%)
- Q4. Average price of laptops released after 1/1/2022 and have at least 2 USB ports.: Success (80%), Partial success (0%), Fail (20%)
- Q5. Average price of laptops released after 1/1/2022 and have at least 2 USB ports grouped by manufacturer.: Success (90%), Partial success (0%), Fail (10%)
- Q6. Average price of laptops with HDD manufactured in an Asian country.: Success (50%), Partial success (0%), Fail (50%)
- Q7. Average price of laptops with HDD manufactured in US.: Success (50%), Partial success (0%), Fail (50%)
- Q8. Count the laptops grouped by the country of the founder.: Success (50%), Partial success (0%), Fail (50%)
- Q9. Average price of laptops grouped by manufacturer.: Success (80%), Partial success (0%), Fail (20%)
- Q10. Average prices of laptops grouped by manufacturer, having average price below 800 Euro.: Success (65%), Partial success (10%), Fail (25%)

We observe that users had higher success rates to questions related to simple aggregations (Q9) and applying filters to the data (Q1–Q5). They were able to successfully navigate the filtering options and effectively apply them to refine and manipulate the dataset. In contrast, they encountered challenges when it came to questions that required formulating paths in the graph data (Q6–Q8). This particular task seemed to present difficulties for users, as they struggled to navigate and comprehend the intricacies of the graph structure. Additionally, users encountered difficulties in understanding how to formulate “HAVING” clauses by loading the results as a new dataset, as they struggled to grasp the concept of applying conditions to aggregated data. This suggests that it is worth improving the system by providing more guidance in such cases (e.g. through info boxes, tooltips, etc.).

The results indicate a clear correlation between participants’ experience levels and their task completion outcomes. Among those classified as “Experts,” a substantial 74% achieved success, while none reported partial success, and 26% faced failures. In the “Intermediate” group, 81.67% marked successful completion, 33.30% reported partial success, and 15% encountered failures. Novice participants demonstrated a 70% success rate, with 2.5% experiencing partial success and 27.5% facing difficulties. These findings highlight the influence of expertise on task completion, with experienced individuals tending to achieve higher success rates, while intermediate participants navigate a balance between success and partial success. The results of the analysis demonstrate that even novice users achieved a commendable level of success, showcasing their ability to perform tasks effectively. Furthermore, it is noteworthy that the differences in task completion rates between novice users and more experienced participants were not significantly pronounced. This suggests that while expertise certainly plays a role in task success, novice users can still perform at a level that is competitive with their more experienced counterparts. These findings underscore the importance of user-friendly interfaces and well-designed systems that can accommodate a diverse range of users, regardless of their experience levels.

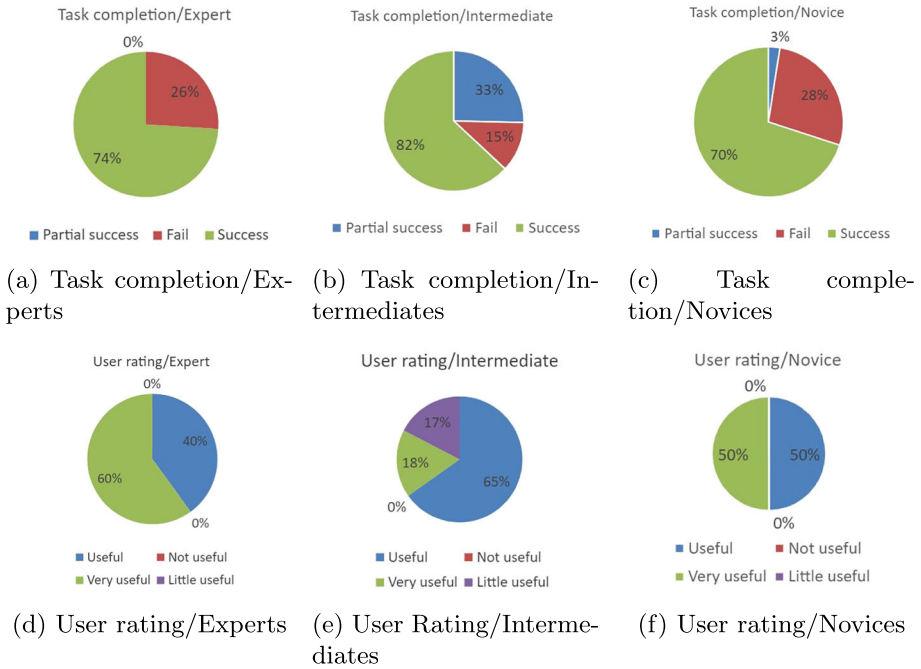


Fig. 16 Task-based evaluation with users: task completion and user rating

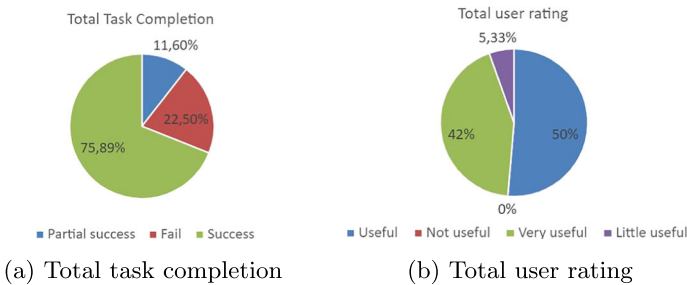


Fig. 17 Task-based evaluation with users: total task completion and total user rating

Regarding *user rating*, among expert users, an impressive 60% found the system “very useful,” indicating a high level of satisfaction and effective utilization. Intermediate users also reported a favorable 16% rating for “very useful,” showcasing its applicability to this user group. Novice users, despite their relative inexperience, expressed a noteworthy 50% rating for “very useful,” highlighting the system’s user-friendly design. The “useful” category received positive feedback from experts and intermediates, with 40% and 60%, respectively, and also garnered a favorable 50% rating from novice users. Remarkably, among expert and intermediate users, the product received no “little useful” or “not useful” ratings, emphasizing its overall effectiveness across these experience levels. These findings underscore the system’s versatility and its ability to cater to users of varying expertise (Fig. 16).

As we can see in Fig. 17a, we can see a substantial total success rate of 75.89%; it is evident that a majority of participants demonstrated proficiency in achieving their objectives.


However, the presence of a partial success rate of 11.60% suggests that there were instances where participants made notable progress, yet encountered some hurdles on their path to success. Meanwhile, a 22.50% fail rate signifies challenges and setbacks faced by participants during their tasks. Also in Fig. 17b, the analysis reveals that the system earned a solid average of 42% for being “Very Useful” across the spectrum of expertise. Additionally, the “Useful” rating garnered an average of 50%, indicating a generally positive reception across all experience levels. It is worth noting that even among less experienced users, represented by the “Novice” category, the system was considered “Very Useful” in 50% of cases, underlining its accessibility and effectiveness. While there was a modest 5.33% rating of “Little Useful” among Intermediate users, there were no instances of a “Not Useful” rating. These results underscore the system’s overall positive reception and suggest that it effectively caters to a diverse range of users, including those with varying levels of expertise.

Overall, the results are very promising in terms of task completion, as shown in Fig. 16a (i.e., success 75.89%, partial success 11.6%, fail 22.5%) and user rating as shown in Fig. 16b (i.e., Very useful 42%, Useful 50%, Little Useful 5,3%, Not Useful 0%), given that no training was provided to the users. These results align well with our target audience, where 55% of users have an expert-level background, 30% are at an intermediate level, and 15% are novice users. It’s important to emphasize that our system is designed with a focus on addressing the needs of non-expert users. Therefore, the positive outcomes in task completion and user satisfaction demonstrate the effectiveness of our system in assisting those who may not have prior expertise.

**User Feedback.** Users provided us with some additional comments for improving our system.

Firstly, there was a call for including a “Reset” button next to the “Analyze” button for clearing their input and starting a fresh with a new analysis.

They suggested to make the “Analyze” button “sticky,” so as to remain visible and accessible as they scroll through the page exploring various facets.

They also highlighted the importance of providing brief explanations next to faceted search symbols, i.e., , for additional guidance.

Additionally, they suggested to enhance the manual with more clear guidelines and possibly sample queries, and screenshots showcasing various usage scenarios.




They faced a difficulty with the concept of “group by” and “group by having a condition,” and they asked for simplification for users not well versed in database queries.

Finally, they suggested to ensure the smooth system availability across different browsers.

In response to user feedback, we implemented several enhancements in our system.

We successfully addressed the call for a more user-friendly interface by introducing a “Reset” button adjacent to the “Analyze” button, allowing users to effortlessly clear their input and start a new analysis.

Furthermore, we have made the “Analyze” button “sticky”, ensuring its visibility and accessibility as users navigate through various facets while scrolling.

To enhance user understanding, we incorporated tooltips next to faceted search symbols (i.e., , , , providing concise explanations for improved guidance.

The manual was enriched with clearer guidelines, sample queries, and screenshots depicting various usage scenarios.

Acknowledging the challenge users faced with the concepts of “group by” and “group by having a condition,” particularly for those less familiar with database queries, we recognize the need for further simplification in these areas and we are exploring ways to present these concepts in a more intuitive manner, ensuring that users of varying levels of expertise can grasp these functionalities.



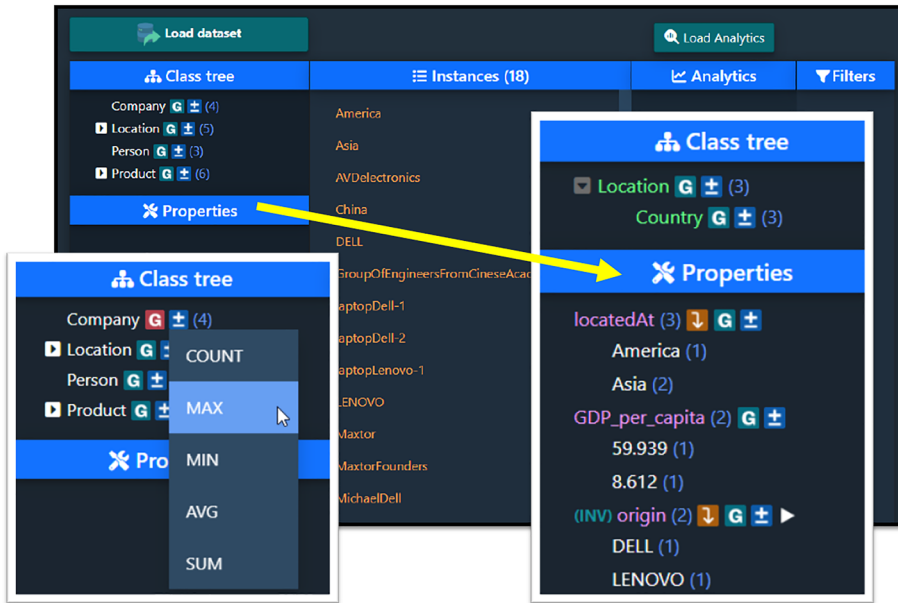


Fig. 18 An alternative implementation of the proposed model

Lastly, in alignment with user suggestions, we are committed to ensuring the smooth availability of our system across different browsers.

### 11.2 Testing implementability

The development of Interactive User Interfaces is in general time-consuming. In this paper, we provided the concrete algorithms for producing the UIs in order to make the model easily implementable. To test the completeness and clarity of the description of the model and the proposed algorithms, we assigned an undergraduate student in the fourth year (not a member of the research group) to implement it (as a Diploma Thesis) by providing him with a preliminary version of the current paper. He was free to decide the implementation technologies he would use. For the back-end side, he used Java, Spring framework, and apache Jena, whereas for the front-end side, he used Vue.js, Bootstrap, and Font Awesome. He managed to implement the model correctly. A few screenshots are shown in Fig. 18.

### 12 Concluding remarks

The formulation of structured queries over RDF Knowledge Graphs is difficult, especially in case that the graph has a broad domain, and thus contains large number of classes and properties. To aid especially novice users (and to save time from expert users), we present a model that aims to enable them to formulate easily analytic queries over any RDF knowledge graph, without having any knowledge of the schema/terminology of the graph, nor the syntax of SPARQL. To come up with an intuitive interaction model and interface, we leverage the familiarity of users with the Faceted Search systems. We start from a general model for

Faceted Search over RDF data, and we extend it with actions that enable users to specify analytic queries, too. Distinctive characteristics of the model are: (1) it can be applied to any RDF dataset (i.e. independently if it follows a star schema), (2) it supports only answerable queries (i.e., it never produces empty results due to lack of data), (3) it supports arbitrarily long paths, (4) it provides count information, (5) it supports the interactive formulation of HAVING clauses, (6) it supports both Faceted Search and analytic queries, and (7) it supports nested analytic queries.

We have detailed the model, specifically (1) we defined formally the state space of the interaction model and the required algorithms for producing the UI (User Interface), (2) we described a hybrid (extensional and intentional) query evaluation approach, (3) we presented an implementation of the model that showcased its feasibility, and (4) we discussed in brief the results of a preliminary evaluation of the proposed system that provided evidence about its acceptance by users in terms of task completion and user rating. As regards the latter, the results of the task-based evaluation with users are very positive in terms of task completion (i.e., Success 75.89%, Partial Success 11.6%, Fail 22.5%) and user rating (Very useful 42%, Useful 50%, Little Useful 5.3%, Not Useful 0%), given that no training was provided to the users, in an audience of 55% expert, 30% intermediate, and 15% novice users. As regards efficiency, without any particular optimization, the system offers real-time interaction, i.e., the analytic queries which are formulated during the interaction require around 3 secs to be evaluated over a KGs with 1 millions triples.

*Perspectives.* The model has wide applicability; it can be applied to any Knowledge Graph; it can complement the other access methods over graph data. In the future, we plan to enrich the model with (1) further visualization features for aiding the interpretation of the analytical results and (2) feature constructor operators (FCO) for cases where data transformations are required. Another direction for future research is to evaluate the system in very large datasets and then investigate what kind of optimizations is required.

**Acknowledgements** Many thanks to Alexandros Perrakis for proof reading the entire paper and for developing the second implementation of the model.

**Author Contributions** All authors contributed to the study conception, design, and writing of this work. The implementation of the system was done by Maria-Evangelia Papadaki.

**Funding** FORTH-ICS.

**Data availability** The dataset used in the running example as well as the running system is publicly accessible.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Consent for publication** Yes.

**Code availability** Upon request to the authors.

## References

1. Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S (2009) DBpedia—a crystallization point for the web of data. *J Web Semant* 7(3):154–165
2. Vrandečić D, Krötzsch M (2014) Wikidata: a free collaborative knowledgebase. *Commun ACM* 57(10):78–85
3. Isaac A, Haslhofer B (2013) Europeana linked open data—data. *Europeana. eu. Semant Web* 4(3):291–297

4. Fafalios P, Petrakis K, Samaritakis G, Doerr K, Kritsotaki A, Tzitzikas Y, Doerr MFASTCAT (2021) collaborative data entry and curation for semantic interoperability in digital humanities. *J Comput Cult Herit (JOCCH)* 14(4):1–20
5. Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z et al (2018) DrugBank 5.0: a major update to the drugbank database for 2018. *Nucl Acids Res* 46(D1):1074–1082
6. Tzitzikas Y, Marketakis Y, Minadakis N, Mountantonakis M, Candela L, Mangiacrapa F et al (2019) Methods and tools for supporting the integration of stocks and fisheries. In: *Information and communication technologies in modern agricultural development: 8th international conference, HAICTA 2017, Chania, Crete, Greece, September 21–24, 2017, Revised Selected Papers 8*. Springer, pp 20–34
7. Koho M, Ikkala E, Leskinen P, Tamper M, Tuominen J, Hyvönen E (2020) Warsampo knowledge graph: Finland in the second world war as linked open data. *Semantic Web—Interoperability, Usability, Applicability*. <https://doi.org/10.3233/SW-200392>. In press
8. Jaradeh MY, Oelen A, Farfar KE, Prinz M, D’Souza J, Kismihók G, Stocker M, Auer S (2019) Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In: *Proceedings of the 10th international conference on knowledge capture*, pp 243–246
9. Dimitrov D, Baran E, Fafalios P, Yu R, Zhu X, Zloch M, Dietze S (2020) TweetsCOVID19—a knowledge base of semantically annotated tweets about the COVID-19 pandemic. In: *Proceedings of the 29th ACM international conference on information & knowledge management*, pp 2991–2998
10. Wang LL, Lo K, Chandrasekhar Y, Reas R, Yang J, Burdick D, Eide D, Funk K, Katsis Y, Kinney R et al (2020) COVID-19 open research dataset (CORD-19). <https://www.kaggle.com/datasets/allen-institute-for-ai/CORD-19-research-challenge>
11. Gazzotti R, Michel FGF (2020) CORD-19 named entities knowledge graph (CORD19-NEKG). Zenodo. <https://doi.org/10.5281/zenodo.3827449>
12. Tzitzikas Y (2022) FS2KG: from file systems to knowledge graphs (demo). In: *ISWC 2022*
13. Mountantonakis M, Tzitzikas Y (2023) Using multiple RDF knowledge graphs for enriching ChatGPT responses. In: *European conference on machine learning and principles and practice of knowledge discovery in databases, ECML PKDD*
14. Chatzakis M, Mountantonakis M, Tzitzikas Y (2021) RDFsim: similarity-based browsing over DBpedia using embeddings. *Information* 12(11):440
15. Nikas C, Kadilierakis G, Fafalios P, Tzitzikas Y (2020) Keyword search over RDF: is a single perspective enough? *Big Data Cogn Comput* 4(3):22
16. Kritsotakis V, Roussakis Y, Patkos T, Theodoridou M (2018) Assistive query building for semantic data. In: *SEMANTICS posters & demos*
17. e Zainab SS, Saleem M, Mehmood Q, Zehra D, Decker S, Hasnain A (2015) FedViz: a visual interface for SPARQL queries formulation and execution. In: *VOILA@ ISWC*, p 49
18. Ferré S (2014) SPARKLIS: a SPARQL endpoint explorer for expressive question answering. In: *ISWC posters and demonstrations track*
19. Akritidis A, Tzitzikas Y (2023) Demonstrating interactive SPARQL formulation through positive and negative examples and feedback. In: *26th international conference on extending database technology, EDBT 2023*
20. Sacco GM, Tzitzikas Y (2009) *Dynamic taxonomies and faceted search: theory, practice, and experience*. Springer, Berlin
21. Tzitzikas Y, Manolis N, Papadakis P (2017) Faceted exploration of RDF/S datasets: a survey. *J Intell Inf Syst* 48(2):329–364
22. Papadaki M-E, Tzitzikas Y (2023) RDF-ANALYTICS: interactive analytics over RDF knowledge graphs. In: *26th international conference on extending database technology, EDBT 2023*
23. Antoniou G, Van Harmelen F (2004) *A semantic web primer*. MIT Press, Cambridge
24. Mountantonakis M, Tzitzikas Y (2018) LODsyndesis: global scale knowledge services. *Heritage* 1(2):23
25. Prieto-Diaz R (1991) Implementing faceted classification for software reuse. *Commun ACM* 34(5):88–97
26. Sacco G (2000) Dynamic taxonomies: a model for large information bases. *IEEE Trans Knowl Data Eng* 12(3):468–479
27. English J, Hearst M, Sinha R, Swearingen K, Yee K-P (2002) Hierarchical faceted metadata in site search interfaces. In: *CHI’02 extended abstracts on human factors in computing systems*, pp 628–639
28. Tunkelang D (2009) *Faceted search*, vol 5. Morgan & Claypool Publishers, San Rafael
29. Russell-Rose T, Tate T (2012) *Designing the search experience: the information architecture of discovery*. Newnes, Oxford, p 45
30. Tessel B (2019) Metadata categorization for identifying search patterns in a digital library. *J Doc* 75(2):270–286. <https://doi.org/10.1108/JD-06-2018-0087>

31. Kobayashi Y, Shindo H, Matsumoto Y (2019) Scientific article search system based on discourse facet representation. *Proc AAAI Conf Artif Intell* 33:9859–9860. <https://doi.org/10.1609/aaai.v33i01.33019859>
32. Moreno-Vega J, Hogan A (2018) GraFa: scalable faceted browsing for RDF graphs. In: *International semantic web conference*. Springer, Berlin, pp 301–317
33. Manioudakis K, Tzitzikas Y (2020) Faceted search with object ranking and answer size constraints. *ACM Trans Inf Syst (TOIS)* 39(1):1–33
34. Arenas M, Grau BC, Kharlamov E, Marciuška Š, Zheleznyakov D (2016) Faceted search over RDF-based knowledge graphs. *J Web Semant* 37:55–74
35. Feddoul L, Schindler S, Löffler F (2019) Automatic facet generation and selection over knowledge graphs. In: *International conference on semantic systems*. Springer, Berlin, pp 310–325
36. Spyrtatos N, Sugibuchi T (2018) HIFUN—a high level functional query language for big data analytics. *J Intell Inf Syst* 51:529–555
37. Papadaki M-E, Tzitzikas Y, Mountantonakis M (2023) A brief survey of methods for analytics over RDF knowledge graphs. *Analytics* 2(1):55–74
38. Ferré S (2021) Analytical queries on vanilla RDF graphs with a guided query builder approach. In: *International conference on flexible query answering systems*. Springer, Berlin, pp 41–53
39. Ferré S (2017) Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. *Semant Web* 8(3):405–418
40. Sherkhonov E, Grau BC, Kharlamov E, Kostylev EV (2017) Semantic faceted search with aggregation and recursion. In: *International semantic web conference*. Springer, Berlin, pp 594–610
41. Kharlamov E, Giacomelli L, Sherkhonov E, Grau BC, Kostylev EV, Horrocks I (2017) Semfacet: making hard faceted search easier. In: *Proceedings of the 2017 ACM on conference on information and knowledge management*, pp 2475–2478
42. Leskinen P, Miyakita G, Koho M, Hyvönen E (2018) Combining faceted search with data-analytic visualizations on top of a SPARQL endpoint. In: *CEUR workshop proceedings*
43. Hyvönen E, Ahola A, Ikkala E (2022) Booksampo fiction literature knowledge graph revisited: building a faceted search interface with seamlessly integrated data-analytic tools. In: *26th international conference on theory and practice of digital libraries, TPDL 2022, Padua, Italy, September 20–23, 2022*. Springer, Berlin, pp 506–511
44. Zhao P, Li X, Xin D, Han J (2011) Graph cube: on warehousing and OLAP multidimensional networks. In: *Proceedings of the 2011 ACM SIGMOD international conference on management of data*, pp 853–864
45. Azirani EA, Goasdoué F, Manolescu I, Roatis A (2015) Efficient OLAP operations for RDF analytics. In: *2015 31st IEEE international conference on data engineering workshops*. IEEE, pp 71–76
46. Benatallah B, Motahari-Nezhad HR et al (2016) Scalable graph-based OLAP analytics over process execution data. *Distrib Parallel Databases* 34:379–423
47. Papadaki M-E, Spyrtatos N, Tzitzikas Y (2021) Towards interactive analytics over RDF graphs. *Algorithms* 14(2):34
48. Hasan SS, Rivera D, Wu X-C, Durbin EB, Christian JB, Tourassi G (2020) Knowledge graph-enabled cancer data analytics. *IEEE J Biomed Health Inform* 24(7):1952–1967
49. Michel F, Gandon F, Ah-Kane V, Bobasheva A, Cabrio E, Corby O, Gazzotti R, Giboin A, Marro S, Mayer T et al (2020) Covid-on-the-Web: knowledge graph and services to advance COVID-19 research. In: *International semantic web conference*. Springer, Berlin, pp 294–310
50. Salast PER, Martin M, Da Mota FM, Auer S, Breitman KK, Casanova MA (2012) Olap2datacube: an ontowiki plug-in for statistical data publishing. In: *2012 second international workshop on developing tools as plug-ins (TOPI)*. IEEE, pp 79–83
51. Zloof MM (1975) Query-by-example: the invocation and definition of tables and forms. In: *Proceedings of the 1st international conference on very large data bases*, pp 1–24
52. Li H, Chan C-Y, Maier D (2015) Query from examples: an iterative, data-driven approach to query construction. *Proc VLDB Endow* 8(13):2158–2169
53. Arenas M, Diaz GI, Kostylev EV (2016) Reverse engineering SPARQL queries. In: *Proceedings of the 25th international conference on world wide web*, pp 239–249
54. Diaz G, Arenas M, Benedikt M (2016) SPARQLByE: querying RDF data by example. *Proc VLDB Endow* 9(13):1533–1536
55. Ali W, Saleem M, Yao B, Hogan A, Ngomo A-CN (2021) A survey of RDF stores & SPARQL engines for querying knowledge graphs. *VLDB J* (2021). (accepted for publication)
56. Nikas C, Fafalios P, Tzitzikas Y (2021) Open domain question answering over knowledge graphs using keyword search, answer type prediction, SPARQL and pre-trained neural models. In: *International semantic web conference*. Springer, Berlin, pp 235–251
57. Ali E, Caputo A, Lawless S, Conlan O (2021) Personalizing type-based facet ranking using BERT embeddings

58. Niu X, Fan X, Zhang T (2019) Understanding faceted search from data science and human factor perspectives. *ACM Trans Inf Syst (TOIS)* 37(2):1–27
59. Tzitzikas Y, Papadaki M-E, Chatzakis M (2021) A spiral-like method to place in the space (and interact with) too many values. *J Intell Inf Syst* 58:1–25
60. Ravindra P, Deshpande VV, Anyanwu K (2010) Towards scalable RDF graph analytics on mapreduce. In: *Proceedings of the 2010 workshop on massive data analytics on the cloud*, pp 1–6
61. Zou L, Özsu MT, Chen L, Shen X, Huang R, Zhao D (2014) gStore: a graph-based SPARQL query engine. *VLDB J* 23:565–590
62. Ibragimov D, Hose K, Pedersen TB, Zimányi E (2015) Processing aggregate queries in a federation of SPARQL endpoints. In: *The semantic web. Latest advances and new domains: 12th European semantic web conference, ESWC 2015, Portoroz, Slovenia, May 31–June 4, 2015. Proceedings 12*. Springer, Berlin, pp 269–285
63. Ibragimov D, Hose K, Pedersen TB, Zimányi E (2016) Optimizing aggregate SPARQL queries using materialized RDF views. In: *The semantic web—ISWC 2016: 15th international semantic web conference, Kobe, Japan, October 17–21, 2016. Proceedings, Part I 15*. Springer, Berlin, pp 341–359
64. Codd EF, Codd SB, Salley CT (1993) Providing OLAP (on-line analytical processing) to user-analysts: an IT mandate. E. F. Codd and Associates
65. Faulkner L (2003) Beyond the five-user assumption: benefits of increased sample sizes in usability testing. *Behav Res Methods Instrum Comput* 35:379–383

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



**Maria-Evangelia Papadaki** obtained her PhD from the Computer Science Department at the University of Crete, Greece, in 2024. She obtained her MSc and BSc from the same department, and she is member of the Information Systems Laboratory (ISL) of FORTH-ICS since 2015. Her research interests fall in the areas of Semantic Web, Linked Open Data, Analytics, and 3D Visualization.



**Yannis Tzitzikas** is Professor of Information Systems in the Computer Science Department at University of Crete (Greece) and Affiliated Researcher of the Information Systems Laboratory at FORTH-ICS (Greece). He studied at the University of Crete and he conducted post-doctoral research at (a) the University of Namur (Belgium), (b) ISTI-CNR (Pisa, Italy), and (c) VTT Technical Research Centre of Finland. Since 2005, he coordinates the Semantic Access and Retrieval group, and from 2019 also the Centre for Cultural Informatics of the Information Systems Laboratory of FORTH-ICS.