# Unifying Faceted Search and Analytics over RDF Knowledge Graphs

Maria-Evangelia Papadaki[1,2*] and Yannis Tzitzikas[1,2]

[1*]Institute of Computer Science, FORTH-ICS, Vasilika Vouton, Heraklion, 70013, Crete, Greece.
[2]Department of Computer Science, University of Crete, Panepistimioupoli Vouton, Heraklion, 70013, Crete, Greece.

*Corresponding author(s). E-mail(s): marpap@ics.forth.gr;
Contributing authors: tzitzik@ics.forth.gr;

**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 the 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. (i) we propose a generic user interface for intuitively analyzing RDF Knowledge Graphs, (ii) we define formally the state-space of the interaction model and the required algorithms for producing the user interface actions, (iii) we present an implementation of the model that showcases its feasibility, and (iv) 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 like KGs like DBpedia [1] and Wikidata [2]), to domain specific repositories (e.g., Europeana [3], DrugBank [4], GRSF [5], ORKG [6], WarSampo [7] and other cultural datasets like [8]), COVID-19 related datasets [9–11], Knowledge Graphs producible from file systems [12], as well as Markup data through `schema.org`,

Plain users can (i) *browse* such graphs (i.e. the user starts from a resource and inspects its values and can move to a connected resource - if dereference URIs are supported), (ii) *search* them using keyword search where the emphasis is on ranking the resources according to relevance (e.g. through multi-perspective keyword search approach [13]), or (iii) use *interactive query formulators* (like A-QuB [14], FedViz [15], SPARKLIS [16], and SPARQL-QBE [17]) However, *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 is considered infeasible for ordinary users, and laborious and time consuming for expert users. 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 would 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.

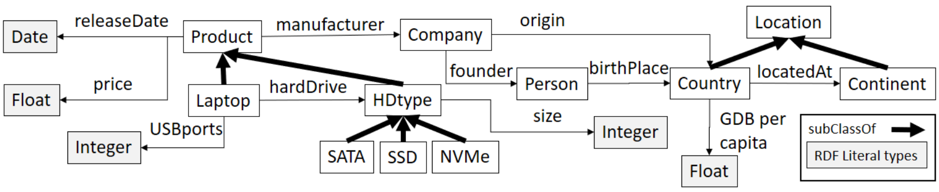


**Figure 1**: The schema of the running example

Consequently, there is a need for an interaction model that will let them formulate such analytic queries with simple clicks, without presupposing knowledge of the vocabulary (schema, ontology, thesauri) and the actual contents of the dataset, or the syntax of the corresponding query language. To

```
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
```

**Figure 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"*.

this end, we leverage the familiarity of users with *Faceted Search* [18], since this model supports the expression of complex conditions with simple clicks. 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 [19]), 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.

The distinctive characteristics of the proposed model are: (i) it can be applied to any RDF dataset (i.e. independently if it follows a star-schema), (ii) it supports only answerable queries (i.e. it never produces empty results due to lack of data), (iii) it supports arbitrarily long paths, (iv) it provides count information, (v) it supports the interactive formulation of HAVING clauses, (vi) it supports nested analytic queries, and (vii) it supports both Faceted Search and analytic queries.

The contributions of this paper are that, (i) we propose a generic user interface for intuitively analyzing RDF Knowledge Graphs without pre-supposing any technical knowledge, (ii) 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, (iii) we present an implementation of the model that showcases its feasibility, and (iv) we discuss the results of an evaluation of the proposed system with users.

The basic idea was demonstrated in the demo paper [20]. In this paper we detail and further elaborate on this direction, specifically: (i) we discuss in more detail the related works and our placement, (ii) we formally specify the interaction model with *states* and *transitions*, (iii) we express the query requirements of the model formally using a QL independent formalism (HIFUN) facilitating in this way the implementation of the model over different technologies, query languages and triplestores; (iv) we provide the exact specification of the UI and the algorithms for facilitating the implementation of the model, (v) we describe the implementation of the model and its evaluation from various perspectives.

The rest of this paper is organized as follows: Section 2 provides the required background. Section 3 describes related work. Section 4 describes the interaction model including the GUI extensions. Section 5 discusses the extensions (in comparison to FS) that are required. Section 6 provides the formal notations and key points of the algorithms, Section 7 presents the algorithm that computes the state space, and Section 8 focuses on how the intentions of the states are expressed and computed. Section 9 discusses an implementation of the model. Section 11 discusses evaluation. Finally, Section 12 concludes the paper and identifies issues for further research.

## 2 Background

Here we briefly discuss RDF (in Section 2.1), Faceted Search (in Section 2.2), and HIFUN (in Section 2.3).

### 2.1 The Resource Description Framework (RDF)

**Resource Description Framework (RDF)** The Resource Description Framework (RDF) [21, 22] 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"). The triples are used for relating Uniform Resource Identifiers (URIs) or anonymous resources (blank nodes) with other URIs, blank nodes or constants (Literals). 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 data set*).

**RDF Schema.** RDF Schema[1] is a special vocabulary which comprises a set of classes with certain properties using the RDF extensible knowledge representation data model. Its intention is to structure RDF resources, since

[1] https://en.wikipedia.org/wiki/RDF_Schema

even though RDF uses URIs to uniquely identify resources, it lacks semantic expressiveness. It uses classes to indicate where a resource belongs, as well as 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 <C rdf:type rdfs:Class> using the predefined class "rdfs:Class" and the predefined property "rdf:type". For example, the triple <ex:Product rdf:type rdfs:Class> indicates that "Product" is a class, while the triple <ex:product1 rdf:type ex:Product> that the individual "*product*1" 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 <ex:manufacturer rdf:type rdf:Property>, <ex:manufacturer rdfs:domain ex:Product>, <ex:manufacturer rdfs:range ex:Company>, 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 <ex:Laptop rdfs:subClassOf ex:Product> denotes that the class "Laptop" is sub-class of the "Product" class. And the triple <ex:fatherOf rdf:subPropertyOf ex:parentOf> defines that the property "fatherOf" is sub-property of "parentOf". In addition, RDFS offers inference functionality[2] as additional information (i.e. discovery of new relationships between resources) about the data it receives. For example, if <ex:myLaptop rdf:type ex:Laptop> and <ex:Laptop rdf:subClassOf ex:Product>, 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*) [23–25] is a widely used interaction scheme for Exploratory Search. It is the de facto query paradigm in e-commerce [18, 26**?** ] and in digital libraries [27, 28]. It is also used for exploring RDF Data (e.g. see [19] for a recent survey, and [29, 30] for recent systems), as well as for exploring general purpose knowledge graphs [31, 32]. There are also recent extensions of the model, with preferences and answer size constraints [30]. 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 results sets.*

[2] https://www.w3.org/standards/semanticweb/inference

## 2.3 HIFUN: A Functional Query Language for Analytics

HIFUN [33] is a high-level functional query language for defining analytic queries over big data sets, independently of how these queries are evaluated. It can be applied over a data set that is structured or unstructured, homogeneous or heterogeneous, centrally stored or distributed. To apply that language over a data set $D$, two assumptions should hold: The data set should i) consist of *uniquely identified data items*, and ii) have a set of *attributes* each of which is viewed as a *function* associating each data item of $D$ with a value, in some set of values. Let $D$ be a data set and $A$ be the set of all attributes ($a_1$, ..., $a_k$) 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 data set $D$ 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 Figure 3. For a more detailed survey, see [34]. In particular, we describe works about formulating analytic queries directly over RDF (in Section 3.1), defining Data Cubes over RDF data (in Section 3.2), defining domain-specific pipelines that produce RDF data and support particular analytic queries (in Section 3.3), and publishing statistical data in RDF (in Section 3.4). At last, we describe our positioning and contribution (in Section 3.5).

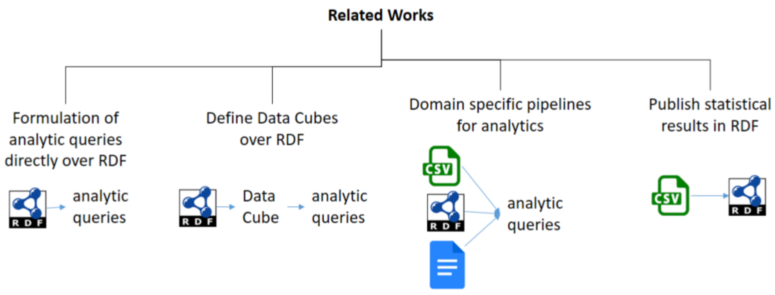

Related Works
Formulation of analytic queries directly over RDF
analytic queries
Define Data Cubes over RDF
Data Cube
analytic queries
Domain specific pipelines for analytics
analytic queries
Publish statistical results in RDF

**Figure 3**: The spectrum of related works

## 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, [35] 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 [36] and it has been adopted in a national French project[3]. 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 [37] that describes a possible extension of SemFacet [38] to support numeric value ranges and aggregation. The authors also state that they work towards 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 are 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 [39] (and the more recent paper [40]), 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] http://data.persee.fr/explore/sparklis/?lang=en

## 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[4] [41–43], and then apply OLAP. One weakness of this approach (as stressed also in [35]) is that is 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 [44] analyzed the applicability of HIFUN (as described in Section 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. However 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 the produce RDF data

There are also, *domain specific works* (focusing on a particular topic, not on any RDF dataset), like [45] that motivates knowledge graph-enabled cancer data analytics. An analogous work for covid-19 related data is [46]. 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. [47]) for publishing statistical data as linked data, and towards this end the RDF data cube vocabulary[5] (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.

[4]https://team.inria.fr/oak/projects/warg/
[5]https://www.w3.org/TR/vocab-data-cube/

## 3.5 Our Position and Contribution

In general, we observe that 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. We present an approach with the following key characteristics: (i) it can be applied to any RDF dataset (i.e. independently if it follows a star-schema), (ii) it supports only answerable queries (i.e. it never produces empty results due to lack of data), (iii) it supports arbitrarily long paths, (iv) it provides count information, (v) it supports the interactive formulation of HAVING clauses, (vi) it supports both Faceted Search and analytic queries, and (vii) it supports nested analytic queries.

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 QL independent formalism facilitating in this way the implementation of the model over different technologies, query languages and triplestores (see [48] 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 Figure 4 (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. $\boxed{\text{G}}$ and $\boxed{\pm}$ next to each facet, as shown in Figure 4 (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 follows the schema of Fig. 1.

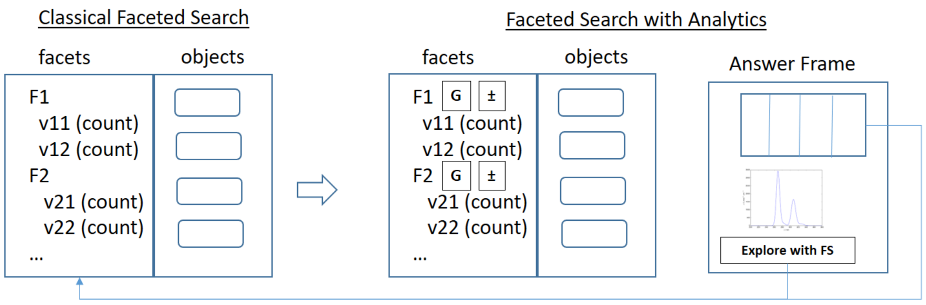


**Figure 4**: The core elements of the GUI for Faceted Search and Analytics

**Example 1 (an AVG query without GROUP BY).** Suppose that, we would like to find *"the average price of laptops that 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 $\boxed{\pm}$ button laid on the right on the "price" facet is offered letting the users click and select the desired function from the displayed menu.

**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 $\boxed{\pm}$ 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 $\boxed{\text{G}}$ 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 $\boxed{\text{G}}$ 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 refer to USB ports. For that, the □ button laid on the right of each facet lets the users filter the values of it fluctuating in a particular range. In this case, the user would click on the □ 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 $\boxed{\text{G}}$ and $\boxed{\pm}$ 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 Figure 5(a). In order to further restrict the answer, we have attached a button, called "Explore with FS" below this table (as shown in Figure 4) which lets the users load the results as a new dataset on the FS system as shown in Figure 5(b). 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.

**(a)**

| | | |
|---|---|---|
| DELL | 2020 | 900 |
| ACER | 2021 | 820 |
| DELL | 2021 | 1000 |
| ACER | 2021 | 850 |

**(b)**

Manufacurers
- DELL (2)
- ACER (2)

Years
- 2020 (2)
- 2021 (2)

Avg Prices
- 820 (1)
- 850 (1)
- 900 (1)
- 1000 (1)

**Figure 5**: Example 3: Exploring the results of an analytic query with faceted search

**Expressing the Queries of the Previous Examples in HIFUN.** Above we described the interaction. During the interaction, HIFUN queries are fomulated. 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: (i) the $\boxed{\text{G}}$ button for grouping the results of the analytical query on this facet and (ii) the $\boxed{\pm}$ 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 $\boxed{\text{G}}$ and $\boxed{\pm}$ buttons.** If the user clicks on the $\boxed{\text{G}}$ 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 $\boxed{\pm}$: If the user clicks on the $\boxed{\pm}$ 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 [44], 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 $\boxed{\text{T}}$, that would let users apply *transformation* functions, i.e. the *feature constructor operators* like those proposed in [44], 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 Section 5.1), and the required extensions of that model for supporting analytics (in Section 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 [19]. 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 to the classical FS system, are that: (i) it leverages the `rdfs:subclassOf` and `rdfs:subPropertyOf` relations, (ii) it supports the formulation of path expressions (exploiting the RDF principles), and (iii) 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 $\boxed{\text{G}}$ and $\boxed{\pm}$ buttons laid next to each facet. In particular,

- clicking on $f.\boxed{\text{G}}$: when the user click on $f.\boxed{\text{G}}$ 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 $f.\boxed{\pm}$: when the user click on $f.\boxed{\pm}$ 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 remain the same, since these buttons do not affect neither the displayed objects on the right frame nor the available transition markers.

Having defined a HIFUN query, the data is analyzed and the results are displayed in tabular format. Note that the result set can be loaded as a new dataset to the system, letting users define analytical queries of unlimited nesting depth over this set.

## 6 The Interaction Model Formally and the Related Algorithms

In Section 6.1, we introduce the notations that we shall use for describing (in Section 6.2) the desired state space of the proposed interaction declaratively (the procedural specification is given later in Section 7).

### 6.1 Notations

**RDF.** Let $K$ be a set of RDF triples and let $C(K)$ 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 $inst(c) = \{o \mid (o, \texttt{rdf:type}, c) \in C(K)\}$ and the instances of a property $p \in Pr$ as $inst(p) = \{ (o,p,o') \mid (o,p,o') \in C(K)\}$.

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 (i) every URI and literal of the dataset i.e., all individuals, [6] or a subset of the dataset, e.g. the result of a keyword query [13], or of a natural language query [49].

### 6.2.1 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 Fig. 6.

Initially, the top-level or maximal classes $maximal_{\leq_{cl}}(C)$ are presented, see Fig. 7 (a). 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 (Fig. 7 (b)) as well as their top-level or maximal properties $maximal_{\leq_{pr}}(Pr)$ (Fig. 7 (c)). 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`

[6] i.e. the results of the SPARQL query "select ?x where { ?x rdf:type owl:NamedIndividual . }"

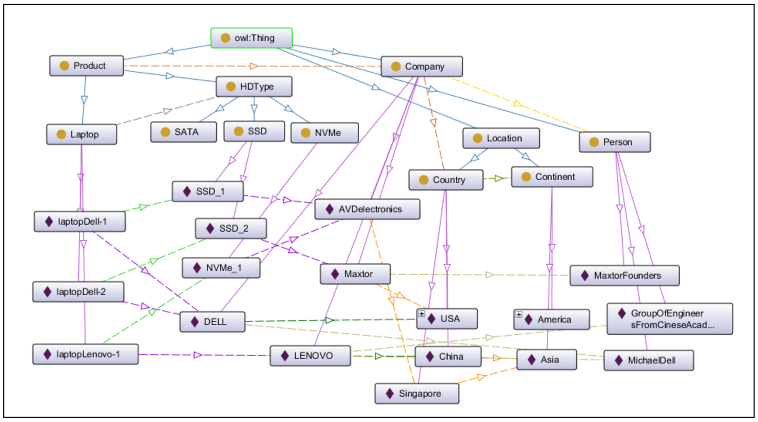


**Figure 6**: Data of our running example

relationships among these classes. Such a layout reflects the structure of the *reflexive and transitive reduction* of the *restriction* of $\leq_{cl}$ on $TM_{cl}(s.Ext)$ (i.e. on $R^{refl,trans}(\leq_{cl} \mid_{TM_{cl}(s.Ext)})$).

### 6.2.2 Property-based transitions

Having expanded the top-level classes, their maximal properties $maximal_{\leq_{pr}}(Pr)$ unfold (Fig. 7 (c)). Each applicable value of these properties corresponds to a "property-based transition marker". Specifically, if $E$ is the current objects, the property-based transitions by $p$ is 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.

**Transitions for Path Expansion.** Let $p_1, \ldots, p_k$ be a sequence of properties. We shall call this sequence *successive*, if $Joins(Joins(\ldots(Joins(s.Ext, p_1), p_2) \ldots p_k) \neq \emptyset$, i.e. if such a sequence does not produce empty results. Let $M_1, \ldots M_k$ 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 $1 \leq i \leq k$, is defined as: $M_i = Joins(M_{i-1}, p_i)$.

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 $M_k, \ldots, M_1$ and finally it will restrict the extension of $s$. Let $M'_k, \ldots M'_1$ 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:

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

**(a)**

- Company (4)
- ⊞ Location (5)
- Person (3)
- ⊞ Product (6)

**(b)**

- Company (4)
- ⊡ Location (5)
  - Continent (2)
  - Country (3)
- Person (3)
- ⊡ Product (6)
  - HDType (3)
    - NVMe (1)
    - SSD (2)
  - Laptop (3)

**(c)**

- by manufacturer (2)
  - DELL (2)
  - Lenovo (1)
- by releaseDate (3)
  - 2021-06-10 (1)
  - 2021-09-03 (1)
  - 2021-10-10 (1)
- by USBports (3)
  - 2 (2)
  - 4 (1)
- by hardDrive (3)
  - SSD1 (1)
  - SSD2 (1)
  - NVMe1 (1)

**(d)**

- by manufacturer (2)
  - DELL (2)
  - Lenovo (1)
- by releaseDate (3)
  - 2021-06-10 (1)
  - 2021-09-03 (1)
  - 2021-10-10 (1)
- by USBports (3)
  - 2 (2)
  - 4 (1)
- by hardDrive (3)
  - *SSD (2)*
    - SSD1 (1)
    - SSD2 (1)
  - *NVMe (1)*
    - NVMe1 (1)

**Figure 7**: (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,

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 each 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 $A_1, \ldots, A_k$, comprising a set of tuples $T = \{t_1, \ldots, t_n\}$. We assign to each tuple $(t_{i1}, \ldots, t_{nk})$ a distinct identifier, say $t_i$, and we produce the following $k$ RDF triples: $(t_i, A_j, t_{ij})$ for each $j = 1 \ldots k$. These $n * k$ triples are loaded to the system and they can be

| **(a)** |
|---|
| by manufacturer (2) |
| DELL (2) |
| Lenovo (1) |
| by releaseDate (3) |
| 2021-06-10 (1) |
| 2021-09-03 (1) |
| 2021-10-10 (1) |
| by USBports (3) |
| 2 (2) |
| 4 (1) |
| by hardDrive (3) |
| SSD1 (1) |
| SSD2 (1) |
| NVMe1 (1) |

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

**Figure 8**: (a): Property-based transition markers, (b): Property Path-based transitions markers

explored as if they were an ordinary RDF dataset. Any restriction expressed over this model, correspond 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 Section 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: (i) from scratch, or (ii) 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 Section 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 ← new Node();
19:        node.name ← $c$.name ;                    ▷ The name that will be displayed
20:        node.count ← $\mid Restrict(s.Ext, c) \mid$ ▷ The count that will accompany the
   name
21:        node.onClick ← 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, 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($E$, "CLASS $c$", $s$).

## 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, Filt: 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 ← 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 ← new Node();
 8:             node.name ← $v$ ;   ▷ The name that will be displayed
 9:             node.count ← $\mid Restrict(s.Ext, p : v) \mid$   ▷ The accompanying count
10:             node.onClick ← ComputeNewState($E$, "PVALUE $p : v$", $s$)
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($\boxed{\text{G}}$, onClick=$gE+=p$)   ▷ For group by
14:         hnode.addButton($\boxed{\pm}$, onClick=$mE+=p$)   ▷ For measuring op
15:         ▷ For Entity Type Switch:
16:         hnode.addButton("EntityTypeSwitch",   ▷ For entity type switch
17:                 onClick ← ComputeNewState($Joins(s.Ext, p)$, $\epsilon$, $s$))
18:         ▷ For Path Expansion:
19:         hnode.addButton("▷",   ▷ For path expansion
20:           onClick ← ComputeNewState($s.Ext$,
21:              "PVALUE p: " + 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:

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

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. $\boxed{\text{G}}$ and $\boxed{\pm}$ are created. Finally, an additional expansion button i.e., "$\triangleright$", is added enabling the user to further expanding the property path (described next in Section 7.4.1).

Clicking on 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($E$, "PVALUE p:v", $s$).

### 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 "$\triangleright$" the algorithm for computing the facets that correspond to path expansion is called, shown in Alg. 3. By clicking on "$\triangleright$" the function "ExpandAndRestrictRecursive(M)" is called again and the process is repeated (as described in Section 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 ← new Node();
 7:             node.name ← $v$ ;              ▷ The name that will be displayed
 8:             node.count ← $| Restrict(M,p:v) |$       ▷ The accompanying count
 9:             node.onClick ← return $Restrict(M,p:v)$  ▷ on click it returns a set
10:         end for
11:         hnode.addButton("▷",              ▷ recursive call for further expansion
12:           onClick ← return $Restrict(M,p,ExpandAndRestrictRecursive(Joins(M,p)))$
13:         hnode.addButton($\boxed{\text{G}}$, onClick=$gE+=p$)               ▷ For group by
14:         hnode.addButton($\boxed{\pm}$, onClick=$mE+=p$)            ▷ For measuring op
15:     end for
16: end function
```

## 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 1 (adapted from [19]) 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.

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

**Table 1**: 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 `temp`

As it is shown, the extension of the current state (i.e. $ctx.Ext$) can be computed either (i) *extensionally* or (ii) *intentionally*. “Extensional” means that the current state is stored in a temporary class `temp` i.e. the RDF triples ($e$, `rdf:type, 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 2.

| 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)” |

**Table 2**: For SPARQL-only evaluation approach

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 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[7] that offers persistent storage and the SPARQL endpoint for executing the queries. The front-end side of the system is implemented in Angular[8]. 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.

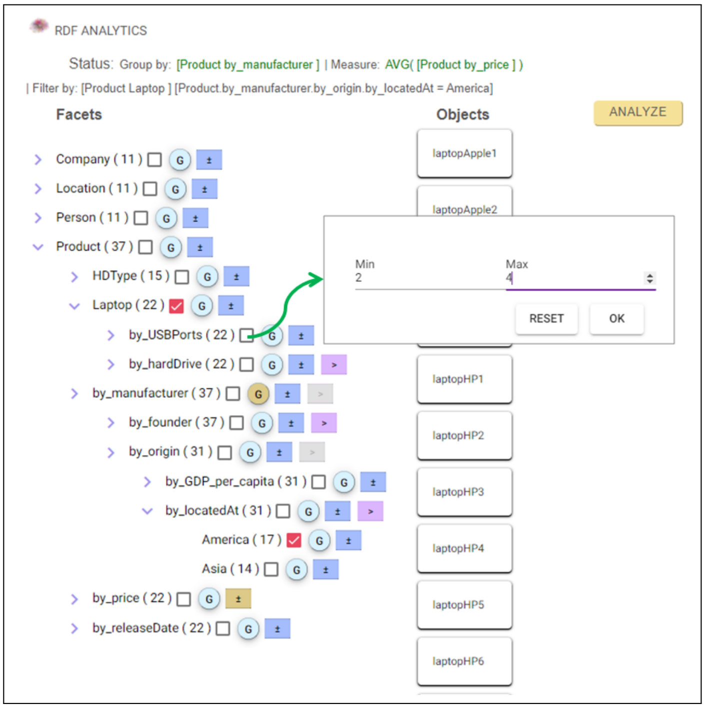
Fig. 9 shows a screenshot of the left frame of the GUI representing the data of our running example. Here, the user has expressed the query "Average price of laptops that have been manufactured by US companies and they have from 2 to 4 USB ports, group by manufacturer". The results of this query are initially displayed in tabular form as shown in Fig. 10 (a). At this point, the user can (i) change the attributes of analysis (by clicking on the "BACK" button), (ii) export the results as a .csv file (by clicking on the "EXPORT AS EXCEL" button), (iii) visualize the results (by clicking on the "VISUALIZE" button) as shown in Fig. 10 (b), or (iv) further restrict the final results (by clicking on the "EXPLORE WITH FS" button) as shown in Fig. 10 (c).

**Common Extensions.** The system could further be enriched with *search boxes* on each individual facet, methods for *facet/value ranking* (as described in the Section 6.3.2 of the survey [19] and more recent ones like [32, 50]), methods for predicting useful facets [51], as well as with similarity based browsing functions (as in [52]). Currently, we are investigating the connection of the system with a 3D visualization (based on [53]) for enabling a more interactive exploration of the results.

### 9.1 Efficiency

The implementation does not have any additional cost except for the evaluation of the respective SPARQL 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 (v) of Table 1) that matches 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 (i) fetches and (ii) counts the range values of the properties that connect to the current resources (as shown in rows (iv) and (vii) of Table 1, respectively),

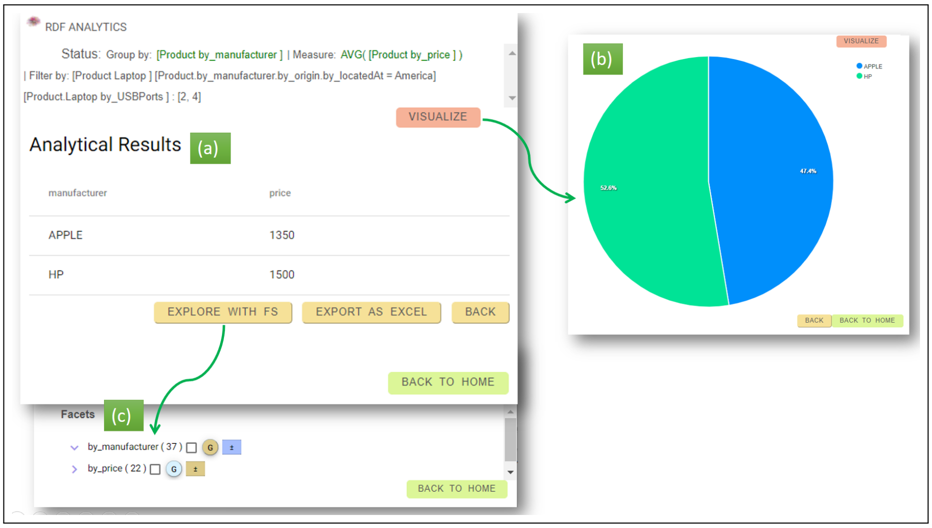
[7]http://docs.openlinksw.com/virtuoso/
[8]https://angular.io/


RDF ANALYTICS

Status: Group by: [Product by_manufacturer ] | Measure: AVG( [Product by_price ] )
| Filter by: [Product Laptop ] [Product.by_manufacturer.by_origin.by_locatedAt = America]

Min: 2 — Max: 4

**Figure 9**: GUI of the system `RDF-ANALYTICS` for formulating the query "Average price of laptops that have 2 to 4 USB ports and have been manufactured by US company, group by manufacturer"


RDF ANALYTICS

Status: Group by: [Product by_manufacturer ] | Measure: AVG( [Product by_price ] )
| Filter by: [Product Laptop ] [Product.by_manufacturer.by_origin.by_locatedAt = America]
[Product.Laptop by_USBPorts ] : [2, 4]

Analytical Results (a)

| manufacturer | price |
|---|---|
| APPLE | 1350 |
| HP | 1500 |

**Figure 10**: Tabular and graphical visualization of the results - Loading of the results as a new dataset.

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 (i) the storing, indexing and query processing techniques of the triplestore (as discussed in [48]) and (ii) the size of the collection. 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, 15000], group by the origin of manufacturer and the birthplace of the founders (complex query containing paths, filters, groups)

**Table 3**: Efficiency

| | Query evaluation (in ms) | | | | | | Presentation of results (in ms) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset size (in triples) | 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 |

As it is shown in Table 3, 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 (in this case Virtuoso) uses, any query optimization is beyond the scope of this work.

## 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 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 Section 10.1), and (b) the OLAP operations it supports (in Section 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 $\boxed{\text{G}}$ 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 $\boxed{\pm}$ button is laid next to each facet name of any level, the system supports both pairing and composition expressions for the measuring function.
- $opE$ the operation expression: it is the aggregation function applied to the grouped values. Since a $\boxed{\pm}$ 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 $\boxed{\text{G}}$ and $\boxed{\pm}$ 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) allows users to view data at different granularities. In order to apply OLAP, data should be organized in a multi-dimensional (MD) structure, known as Data Cube. A Data Cube consists of (i) facts which are the subjects of the analysis and quantified by measures, and (ii) 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 proposed system, supports all of them. In particular, traversing up the hierarchy of a facet corresponds to a roll-up operation, traversing down the hierarchy of a facet 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.

## 11 Evaluation

Section 11.1 compares the proposed model with the most relevant systems for RDF analytics, Section 11.2 discusses the results of an evaluation of our approach with users and Section 11.3 conveys an implementation of the model that proves its feasibility and the completeness of the introduced algorithms.

### 11.1 Comparison with Related Systems

In Table 4 we qualify the more relevant systems, mentioned in Section 3, 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 to the related systems, `RDF-ANALYTICS` outstands since (i) it can be applied to any RDF graphs without requiring data pre-processing, (ii) it supports plain browsing of the graph as well as analysis

**Table 4**: 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 |
|---|---|---|---|---|---|---|---|---|
| [37] | ANY | Yes | Yes | Yes but with No Count information | Not explicitly, reachability | No | No | No |
| [35] | ANY | Yes | No | No. Special interface | Not clear | No | Yes | Yes |
| [39] | ANY | Yes | No | Yes | Yes with counts | Yes | Yes | No |
| Our approach | ANY | Yes | Yes by AF | Yes | Yes with counts | Yes | Yes | Yes |

of it, (iii) it supports restrictions of the results and property paths, (iv) it offers visualization of the results in a running system that is publicly available and (v) it has been evaluated by users

## 11.2 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 [54], 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%)), 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[9].

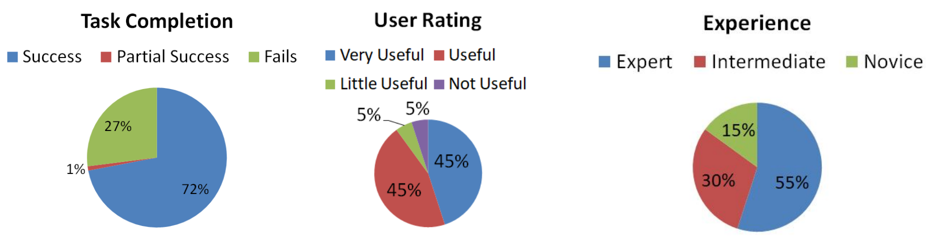
We defined 10 tasks for the evaluation, below we list them along with the success rates of the users.

[9]The deployment of the system that was used is accessible at https://demos.isl.ics.forth.gr/rdf-analytics.

- 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.).

Overall, the results are very promising in terms of task completion, as shown in Fig. 11 (a) (i.e., success 72%, partial success 1%, fail 27%) and user rating as shown in Fig. 11 (b) (i.e., Very useful 45%, Useful 45%, Little Useful 5%, Not Useful 5%), 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.



(a) Task completion (b) User Rating (c) User Experience

**Figure 11**: Task-based evaluation with users: task completion and user rating

In future, we plan to improve the system based on the users' feedback and to enrich the system with additional features, especially regarding the visualization of the results.

## 11.3 Testing Implementability

The development of Interactive User Interfaces are 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 Figure 12.

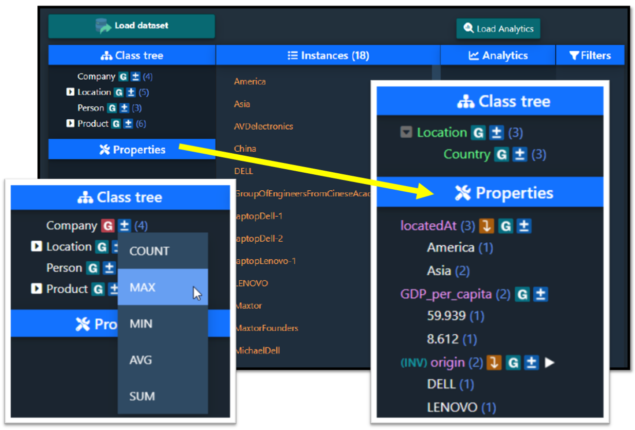


**Figure 12**: An alternative implementation of the proposed model

## 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: (i) it can be applied to any RDF dataset (i.e. independently if it follows a star-schema), (ii) it supports only answerable queries (i.e. it never produces empty results due to lack of data), (iii) it supports arbitrarily long paths, (iv) it provides count information, (v) it supports the interactive formulation of HAVING clauses, (vi) it supports both Faceted Search and analytic queries, and (vii) it supports nested analytic queries.

We detail the model, specifically (i) we define formally the state-space of the interaction model and the required algorithms for producing the UI (User Interface) (ii) we describe a hybrid (extensional and intentional) query evaluation approach, (iii) we present an implementation of the model that showcases its feasibility, and (iv) we discuss in brief the results of a preliminary evaluation of the proposed system that provides evidence about its acceptance by users. In the future, we plan to enrich the model with (i) further visualization features for aiding the interpretation of the analytical results and (ii) feature constructor operators (FCO) for cases where data transformations are required. Another direction for future research is to investigate methods for improving the efficiency of the proposed model, both the the formulation and execution of the analytical queries.

## Acknowledgments

Many thanks to Alexandros Perrakis for proof-reading the paper and its algorithms and for the second implementation of the model.

## Statements and Declarations

• Funding: FORTH-ICS
• Conflict of interest/Competing interests: None
• Ethics approval: Not Applicable
• Consent for publication: Yes
• Availability of data and materials: The dataset used in the running example, as well as the running system is publicly accessible.
• Code availability: Upon request to the authors
• Authors' 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.

## References

[1] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia-a crystallization point for the web of data. Journal of web semantics **7**(3), 154–165 (2009)

[2] Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)

[3] Isaac, A., Haslhofer, B.: Europeana linked open data–data. europeana. eu. Semantic Web **4** (2013)

[4] Wishart, D.S., Feunang, Y.D., Guo, A.C., Lo, E.J., Marcu, A., Grant, J.R., Sajed, T., Johnson, D., Li, C., Sayeeda, Z., *et al.*: Drugbank 5.0: a major update to the drugbank database for 2018. Nucleic acids research **46**(D1), 1074–1082 (2018)

[5] Tzitzikas, Y., Marketakis, Y., Minadakis, N., Mountantonakis, M., Candela, L., Mangiacrapa, F., *et al.*: Methods and tools for supporting the integration of stocks and fisheries. In: Chapter in Information and Communication Technologies in Modern Agricultural Development, Springer, 2019. (2019). Springer

[6] Jaradeh, M.Y., Oelen, A., Farfar, K.E., Prinz, M., D'Souza, J., Kismihók, G., Stocker, M., Auer, S.: Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 243–246 (2019)

[7] Koho, M., Ikkala, E., Leskinen, P., Tamper, M., Tuominen, J., Hyvönen, E.: Warsampo knowledge graph: Finland in the second world war as linked open data. Semantic Web – Interoperability, Usability, Applicability (2020). https://doi.org/10.3233/SW-200392. In press

[8] Fafalios, P., Petrakis, K., Samaritakis, G., Doerr, K., Kritsotaki, A., Tzitzikas, Y., Doerr, M.: Fast cat: Collaborative data entry and curation for semantic interoperability in digital humanities. ACM Journal on Computing and Cultural Heritage **14** (2021)

[9] Dimitrov, D., Baran, E., Fafalios, P., Yu, R., Zhu, X., Zloch, M., Dietze, S.: Tweetscov19–a knowledge base of semantically annotated tweets about the covid-19 pandemic. In: 29th ACM International Conference on Information and Knowledge Management (CIKM 2020) (2020)

[10] Covid-19 open research dataset (cord-19). (2020)

[11] R. Gazzotti, F.G. F. Michel: Cord-19 named entities knowledge graph (cord19-nekg). (2020)

[12] Tzitzikas, Y.: FS2KG: From file systems to knowledge graphs (demo). In: ISWC 2022 (2022)

[13] Nikas, C., Kadilierakis, G., Fafalios, P., Tzitzikas, Y.: Keyword search over rdf: Is a single perspective enough? Big Data and Cognitive Computing **4**(3), 22 (2020)

[14] Kritsotakis, V., Roussakis, Y., Patkos, T., Theodoridou, M.: Assistive query building for semantic data. In: SEMANTICS Posters&Demos (2018)

[15] e Zainab, S.S., Saleem, M., Mehmood, Q., Zehra, D., Decker, S., Hasnain, A.: Fedviz: A visual interface for sparql queries formulation and execution. In: VOILA@ ISWC, p. 49 (2015)

[16] Ferré, S.: Sparklis: a sparql endpoint explorer for expressive question answering. In: ISWC Posters & Demonstrations Track (2014)

[17] Akritidis, A., Tzitzikas, Y.: Demonstrating interactive SPARQL formulation through positive and negative examples and feedback. In: 26th International Conference on Extending Database Technology, EDBT 2023 (2023)

[18] Sacco, G.M., Tzitzikas, Y., *et al.*: Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience. Springer, ??? (2009)

[19] Tzitzikas, Y., Manolis, N., Papadakos, P.: Faceted exploration of RDF/S datasets: a survey. Journal of Intelligent Information Systems **48**(2), 329–364 (2017)

[20] Papadaki, M.-E., Tzitzikas, Y.: Rdf-analytics: Interactive analytics over rdf knowledge graphs. In: 26th International Conference on Extending Database Technology, EDBT 2023 (2023)

[21] Antoniou, G., Van Harmelen, F.: A Semantic Web Primer. MIT Press, ??? (2004)

[22] Mountantonakis, M., Tzitzikas, Y.: LODsyndesis: Global scale knowledge services. Heritage **1** (2018)

[23] Prieto-Diaz, R.: Implementing faceted classification for software reuse. Communications of the ACM **34**(5), 88–97 (1991)

[24] Sacco, G.: Dynamic taxonomies: A model for large information bases. IEEE Transactions on Knowledge and Data Engineering **12**(3), 468–479 (2000)

[25] English, J., Hearst, M., Sinha, R., Swearingen, K., Yee, K.-P.: Hierarchical faceted metadata in site search interfaces. In: CHI'02 Extended Abstracts on Human Factors in Computing Systems, pp. 628–639 (2002)

[26] Tunkelang, D.: Faceted search. Synthesis lectures on information concepts, retrieval, and services **1**(1), 1–80 (2009)

[27] Tessel, B.: Metadata categorization for identifying search patterns in a digital library **75**(2), 270–286 (2019). https://doi.org/10.1108/JD-06-2018-0087

[28] Kobayashi, Y., Shindo, H., Matsumoto, Y.: Scientific article search system based on discourse facet representation. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 9859–9860 (2019). https://doi.org/10.1609/aaai.v33i01.33019859

[29] Moreno-Vega, J., Hogan, A.: Grafa: Scalable faceted browsing for RDF graphs. In: International Semantic Web Conference, pp. 301–317 (2018). Springer

[30] Manioudakis, K., Tzitzikas, Y.: Faceted search with object ranking and answer size constraints. ACM Transactions on Information Systems (TOIS) **39**(1), 1–33 (2020)

[31] Arenas, M., Grau, B.C., Kharlamov, E., Marciuška, Š., Zheleznyakov, D.: Faceted search over RDF-based knowledge graphs. Journal of Web Semantics **37**, 55–74 (2016)

[32] Feddoul, L., Schindler, S., Löffler, F.: Automatic facet generation and selection over knowledge graphs. In: International Conference on Semantic Systems, pp. 310–325 (2019). Springer

[33] Spyratos, N., Sugibuchi, T.: HIFUN-a high level functional query language for big data analytics. Journal of Intelligent Information Systems **51** (2018)

[34] Papadaki, M.-E., Tzitzikas, Y., Mountantonakis, M.: A brief survey of methods for analytics over rdf knowledge graphs. Analytics **2**(1), 55–74 (2023)

[35] Ferré, S.: Analytical queries on vanilla RDF graphs with a guided query builder approach. In: International Conference on Flexible Query Answering Systems, pp. 41–53 (2021). Springer

[36] Ferré, S.: Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language. Semantic Web **8**(3), 405–418 (2017)

[37] Sherkhonov, E., Grau, B.C., Kharlamov, E., Kostylev, E.V.: Semantic faceted search with aggregation and recursion. In: International Semantic Web Conference, pp. 594–610 (2017). Springer

[38] Kharlamov, E., Giacomelli, L., Sherkhonov, E., Grau, B.C., Kostylev, E.V., Horrocks, I.: Semfacet: Making hard faceted search easier. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2475–2478 (2017)

[39] Leskinen, P., Miyakita, G., Koho, M., Hyvönen, E.: Combining faceted search with data-analytic visualizations on top of a sparql endpoint. In: CEUR Workshop Proceedings (2018)

[40] Hyvönen, E., Ahola, A., Ikkala, E.: 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, pp. 506–511 (2022). Springer

[41] Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and olap multidimensional networks. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (2011)

[42] Azirani, E.A., Goasdoué, F., Manolescu, I., Roatiş, A.: Efficient OLAP operations for RDF analytics. In: 2015 31st IEEE International Conference on Data Engineering Workshops, pp. 71–76 (2015). IEEE

[43] Benatallah, B., Motahari-Nezhad, H.R., et al.: Scalable graph-based olap analytics over process execution data. Distributed and Parallel Databases **34** (2016)

[44] Papadaki, M.-E., Spyratos, N., Tzitzikas, Y.: Towards interactive analytics over RDF graphs. Algorithms **14**(2), 34 (2021)

[45] Hasan, S.S., Rivera, D., Wu, X.-C., Durbin, E.B., Christian, J.B., Tourassi, G.: Knowledge graph-enabled cancer data analytics. IEEE journal of biomedical and health informatics **24**(7), 1952–1967 (2020)

[46] Michel, F., Gandon, F., Ah-Kane, V., Bobasheva, A., Cabrio, E., Corby, O., Gazzotti, R., Giboin, A., Marro, S., Mayer, T., *et al.*: Covid-on-the-web: Knowledge graph and services to advance covid-19 research. In: International Semantic Web Conference, pp. 294–310 (2020). Springer

[47] Salast, P.E.R., Martin, M., Da Mota, F.M., Auer, S., Breitman, K.K.,

Casanova, M.A.: Olap2datacube: An ontowiki plug-in for statistical data publishing. In: 2012 Second International Workshop on Developing Tools as Plug-Ins (TOPI), pp. 79–83 (2012). IEEE

[48] Ali, W., Saleem, M., Yao, B., Hogan, A., Ngomo, A.-C.N.: A survey of RDF stores & SPARQL engines for querying knowledge graphs. VLDB Journal (2021). (accepted for publication)

[49] Nikas, C., Fafalios, P., Tzitzikas, Y.: Open domain question answering over knowledge graphs using keyword search, answer type prediction, SPARQL and pre-trained neural models. In: International Semantic Web Conference, pp. 235–251 (2021). Springer

[50] Ali, E., Caputo, A., Lawless, S., Conlan, O.: Personalizing type-based facet ranking using bert embeddings (2021)

[51] Niu, X., Fan, X., Zhang, T.: Understanding faceted search from data science and human factor perspectives. ACM Transactions on Information Systems (TOIS) **37**(2), 1–27 (2019)

[52] Chatzakis, M., Mountantonakis, M., Tzitzikas, Y.: RDFsim: Similarity-Based Browsing over DBpedia Using Embeddings. Information **12**(11), 440 (2021)

[53] Tzitzikas, Y., Papadaki, M.-E., Chatzakis, M.: A spiral-like method to place in the space (and interact with) too many values. Journal of Intelligent Information Systems, 1–25 (2021)

[54] Faulkner, L.: Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. Behavior Research Methods, Instruments, & Computers **35**, 379–383 (2003)