

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCES AND ENGINEERING

Interactive Analytics over RDF Knowledge Graphs

by

Maria-Evangelia Papadaki

PhD Dissertation

Presented

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

Heraklion, December 2023

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
Interactive Analytics over RDF Knowledge Graphs

PhD Dissertation Presented
by **Maria-Evangelia Papadaki**
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

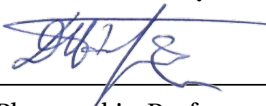
APPROVED BY:



Author: Maria-Evangelia Papadaki



Supervisor: Yannis Tzitzikas, Professor, University of Crete



Committee Member: Dimitrios Plexousakis, Professor, University of Crete and Director, ICS-FORTH



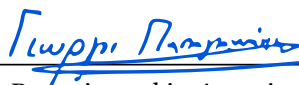
Committee Member: Nikolaos Spyrtatos, Professor, University of Paris-South



Committee Member: Grigorios Antoniou, Professor, University of Huddersfield



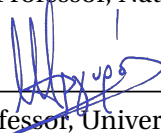
Committee Member: Kostas Magoutis, Associate Professor, University of Crete



Committee Member: George Papagiannakis, Associate Professor, University of Crete



Committee Member: Christos Papatheodorou, Professor, National and Kapodistrian University of Athens



Department Chairman: Antonis Argyros, Professor, University of Crete

Heraklion, December 2023

To the boundless realms of the freedom of mind.

Abstract

Today, numerous Knowledge Graphs, expressed in RDF, play a crucial role in consolidating and integrating data from diverse sources. It would be very valuable to delve into the analysis of these graphs for enhanced insights and understanding. However, formulating analytical queries over Knowledge Graphs in RDF is a challenging task due to the complexity and scale of these graphs that presupposes familiarity with the syntax of the corresponding query language (i.e. SPARQL) and the contents of the graph. To alleviate this problem, we introduce an interactive model to assist users in formulating analytic queries over complex RDF Knowledge Graphs, irrespective of their schema structure. This is particularly crucial, since in non-star-schema-based knowledge graphs, the presence of non-star-schema relationships requires a more complex querying approach. To provide an intuitive interface, we leverage users' familiarity with Faceted Search systems, and we extend it for enabling the formulation of analytic queries in a user-friendly way. In particular, we start from a general model for Faceted Search over RDF data, and we extend it with actions that empower users to formulate simple and complex analytic queries, as well. These actions correspond to queries of a high-level query language for analytics, named HIFUN, that we then translate to SPARQL. Most, the proposed interactive model serves a dual purpose, addressing not only the formulation of analytic queries, but also the formulation of exploratory queries; it lets users transition seamlessly from locating resources in a Faceted Search manner to performing in-depth analyses of the underlying RDF Knowledge Graph. This accommodates the diverse needs of users, enabling both flexible and dynamic exploration and analysis of the graph. Additionally, the formulation of queries, including nested ones, is gradual acknowledging the iterative nature of data analysis. This process involves repeated and refining steps, allowing users to deepen their queries as they gain insights into the graph's structure and content. Overall, the main contributions of this dissertation are: (i) we present a user-friendly interface for intuitively analyzing RDF Knowledge Graphs, and (ii) we formally define the state-space of the interaction model as well as the algorithms needed to produce user interface actions. We also describe and provide a complete implementation of the model and the relating algorithms, showcasing its feasibility in real-world scenarios. This emphasizes the practical applicability of our approach, making it valuable both for analysts and ordinary users dealing with RDF Knowledge Graphs. Finally, we discuss the results of a user evaluation, providing evidence of the method's acceptance. This empirical validation not only underscores the effectiveness of our model, but also sheds light on future development. In essence, our research not only tackles the

complexities of formulating analytic queries over RDF Knowledge Graphs, but also emphasizes the friendliness and acceptance by users.

Keywords: Knowledge Graphs, Analytics, Faceted Search

Supervisor: Yannis Tzitzikas
Professor
Computer Science Department
University of Crete

Περίληψη

Σήμερα, πολλοί Γνωσιακοί Γράφοι, εκφρασμένοι σε RDF, διαδραματίζουν κρίσιμο ρόλο στην συγκέντρωση και ενοποίηση δεδομένων από διαφορετικές πηγές. Θα ήταν πολύτιμο να μπορούμε να εμβαθύνουμε στην ανάλυση αυτών των γράφων για να ενισχύσουμε την κατανόησή μας και την εξαγωγή συμπερασμάτων. Ωστόσο, η διατύπωση αναλυτικών επερωτήσεων σε Γράφους Γνώσης RDF είναι δύσκολη αφού προϋποθέτει εξοικείωση και με το συντακτικό των αντίστοιχων γλωσσών επερωτήσεως (ήτοι την SPARQL) και με τα περιεχόμενα του Γνωσιακού Γράφου. Για να απαλύνουμε αυτό το πρόβλημα, προτείνουμε ένα διαδραστικό μοντέλο για να βοηθήσουμε τους χρήστες να διατυπώνουν αναλυτικά ερωτήματα σε πολύπλοκους RDF Γράφους Γνώσης, ανεξάρτητα από τη δομή του σχήματός τους. Αυτό είναι ιδιαίτερα σημαντικό, καθώς σε γνωσιακούς γράφους που δεν βασίζονται σε Star σχήματα, η παρουσία περίπλοκων συνδέσεων απαιτεί μια πιο σύνθετη προσέγγιση διατύπωσης επερωτήσεων. Για να προσφέρουμε μια διαισθητική διεπαφή, αξιοποιούμε την εξοικείωση των χρηστών με τα συστήματα Πολύπλευρης Αναζήτησης (Faceted Search), επεκτείνοντας ένα τέτοιο μοντέλο που θα προσφέρει και δυνατότητες ανάλυσης με έναν προσιτό και φιλικό προς τον χρήστη τρόπο. Συγκεκριμένα, ξεκινώντας από ένα γενικό μοντέλο για Πολύπλευρη Αναζήτηση επί δεδομένων RDF, το επεκτείνουμε με ενέργειες που δίνουν τη δυνατότητα στους χρήστες να διατυπώνουν και σύνθετα αναλυτικά ερωτήματα. Αυτές οι ενέργειες αντιστοιχούν σε επερωτήσεις μιας γλώσσας επερωτήσεων υψηλού επιπέδου για ανάλυση δεδομένων, που ονομάζεται HIFUN, τις οποίες κατόπιν μεταφράζουμε σε SPARQL. Μάλιστα, το προτεινόμενο διαδραστικό μοντέλο εξυπηρετεί διττό σκοπό, απευθυνόμενο όχι μόνο στη διατύπωση αναλυτικών ερωτημάτων, αλλά και στην εξερεύνηση των δεδομένων αφού επιτρέπει στους χρήστες να μεταβαίνουν απρόσκοπτα από τον εντοπισμό πόρων με τρόπο Πολύπλευρης Αναζήτησης στην εκτέλεση εις βάθος αναλύσεων του υποκείμενου Γράφου Γνώσης RDF. Αυτό καλύπτει τις διαφορετικές ανάγκες των χρηστών, προσφέροντας ευέλικτη και δυναμική εξερεύνηση και ανάλυση του γράφου. Επιπροσθέτως, η διατύπωση ερωτημάτων, συμπεριλαμβανομένων των εμφωλευμένων, είναι σταδιακή αναγνωρίζοντας τον επαναληπτικό χαρακτήρα της ανάλυσης δεδομένων. Αυτή η διαδικασία περιλαμβάνει επαναλαμβανόμενα βήματα εκλέπτυνσης, επιτρέποντας στους χρήστες να εμβαθύνουν τα ερωτήματά τους καθώς αποκτούν πληροφορίες για τη δομή και το περιεχόμενο του γράφου. Συνολικά, οι κύριες συνεισφορές της διατριβής αυτής είναι: (i) προτείνουμε φιλική προς το χρήστη διεπαφή για διαισθητική ανάλυση Γρά-

φων Γνώσης RDF και, (ii) ορίζουμε τυπικά τον χώρο καταστάσεων του μοντέλου αλληλεπίδρασης καθώς και τους αλγόριθμους που απαιτούνται για την παραγωγή ενεργειών για τη διεπαφή χρήσης. Παρέχουμε επίσης μια λεπτομερή περιγραφή και πλήρη υλοποίηση του μοντέλου και των σχετικών αλγορίθμων, καταδεικνύοντας τη εφικτότητά του σε σενάρια πραγματικού κόσμου. Αυτό τονίζει την πρακτική εφαρμογή της προσέγγισής μας, καθιστώντας την πολύτιμη τόσο για τους αναλυτές όσο και για τους απλούς χρήστες που ασχολούνται με Γράφους Γνώσης RDF. Τέλος, σχολιάζουμε την αξιολόγηση του συστήματος από χρήστες, της οποίας τα αποτελέσματα ήταν πολύ θετικά αναφορικά με την αποδοχή και την αποτελεσματικότητα της μεθόδου. Αυτή η εμπειρική επικύρωση όχι μόνο τονίζει την αποτελεσματικότητα του μοντέλου μας, αλλά έδωσε και ανατροφοδότηση για τη μελλοντική επέκτασή του. Ουσιαστικά, η έρευνά μας όχι μόνο αντιμετωπίζει την πολυπλοκότητα της διατύπωσης αναλυτικών ερωτημάτων σε Γράφους Γνώσης RDF, αλλά δίνει επίσης έμφαση στην αποδοχή των χρηστών.

Λέξεις κλειδιά: Γράφοι Γνώσης, Ανάλυση, Πολύπλευρη Αναζήτηση

Επόπτης: Τζιτζικας Γιάννης
Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Contents

Abstract	vii
Περίληψη (Abstract in Greek)	ix
Table of Contents	xi
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 General Objective	1
1.2 Motivation and Vision	1
1.3 Research Question	3
1.4 The Approach	3
1.5 Contributions of this Dissertation	5
1.5.1 Publications produced in the context of this dissertation	5
1.6 Outline of Dissertation	7
2 Background	9
2.1 The Resource Description Framework (RDF)	9
2.2 The Spectrum of Access Methods over RDF	11
2.3 Analytical Query	13
2.4 SPARQL Protocol and RDF Query Language (SPARQL)	13
2.5 HIFUN: A Functional Query Language for Analytics	14
2.5.1 Using HIFUN as an Interface to RDF Dataset	18
2.6 Faceted Search	18
3 Related Work	21
3.1 Past Surveys: Integration, Querying and Visualization of RDF Knowledge Graphs	21
3.2 RDF and Analytics: Challenges and General Approaches	22
3.2.1 Challenges	22
3.2.2 Categories of Works (related to RDF and Analytics)	22
3.2.3 Categories of Analytic Queries	22
3.3 Survey of works and Systems	24
3.3.1 Methodology and Statistics	24
3.3.2 C1. Formulation of Analytic Queries directly over RDF	25
3.3.3 C2. Definition of Data Cubes over RDF	28
3.3.4 C3. Domain-specific Pipelines over RDF	32

3.3.5	C4. Publishing of Statistical Data in RDF	33
3.3.6	C5. Quality Analytics Over Multiple RDF Datasets	35
3.4	Efficiency and Visualization	37
3.4.1	Efficiency	37
3.4.2	Visualization of Results	39
3.5	Summary	39
3.6	Our Positioning and Focus	40
4	On Applying HIFUN over RDF	43
4.1	Applicability of HIFUN over RDF	43
4.1.1	Prerequisites for Applying HIFUN over RDF Data	43
4.1.2	Methods to Apply HIFUN over RDF	44
4.2	Translation of HIFUN Queries to SPARQL	46
4.2.1	Simple Queries	46
4.2.2	Attribute-Restricted Queries.	47
4.2.3	Results-Restricted Queries.	48
4.2.4	Complex Grouping Queries.	48
4.2.5	The Full Algorithm for Translating a HIFUN Query to a SPARQL Query	50
4.2.6	Cases where the Prerequisites of HIFUN are not Satisfied	53
5	The Proposed Interaction Model for Analytics over RDF	59
5.1	The Interaction Model in Brief	59
5.2	The Required Extensions of the Formal Model for FS over RDF for supporting Analytics	64
5.2.1	Background: The Core Model for FS over RDF	64
5.2.2	The Extension of the Model for Analytics (Formally)	64
5.3	The Interaction Model Formally and the Related Algorithms	65
5.3.1	Notations	65
5.3.2	Defining the State Space of the Interaction	66
5.3.3	Loading AF as a new Dataset	68
5.4	The Algorithm that Implements the State Space	69
5.4.1	Starting Points	69
5.4.2	Computing the Objects in the Right Frame	69
5.4.3	Computing the Facets corresponding to Classes	70
5.4.4	Computing the Facets that correspond to Properties	71
5.5	Expressing and Computing the Intentions of the States	72
6	Implementation	75
6.1	Architecture	75
6.2	System Demonstration	76
6.3	3D Visualization	77
6.4	Efficiency	80

7	The Expressive Power of the Model	85
7.1	Expressible HIFUN queries	85
7.2	OLAP Operators Supported	86
8	Evaluation	91
8.1	Task-based Evaluation with Users	91
8.2	Testing Implementability	95
9	Conclusion	97
9.1	Future Work and Research	98
	Bibliography	101

List of Figures

1.1	From Disparate and Fragmented Datasets to Knowledge Graphs	2
1.2	The schema of the running example	3
1.3	Expression in SPARQL of the query “ <i>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</i> ”.	4
2.1	Context and main elements	9
2.2	Example of an RDF triple	10
2.3	Example of an RDF graph	10
2.4	Example of an RDFS schema about products	11
2.5	An Overview of the Access Methods over RDF	13
2.6	Expression in SPARQL of the query “ <i>total quantities of products released by company</i> ”.	14
2.7	Representation of Invoices Dataset in HIFUN	15
2.8	An analytic query and its answer	16
2.9	Example of an Analysis Context in HIFUN	17
2.10	Faceted search system	19
2.11	Examples of Faceted Search Systems	20
3.1	The spectrum of related works in 5 different categories	23
3.2	The number of surveyed works per category	24
3.3	The publication year of the surveyed papers	24
3.4	Indicative Screenshots of Visualization of analytical results for Categories C1-C2: (a) column chart (C1, C2), (b) bar chart (C2), (c) line chart (C2), (d) pie chart (C2), (e) bubble chart (C2), (f) geo chart (C2), (g) area (C2), (h) treemap (C1, C2), (i) graph (C2), (j) table (C2)	38
3.5	Indicative Screenshots of Visualization of analytical results for Categories C3-C5: (a) graph chart (C3), (b) pie chart (C3), (c,d) bar charts (C4,C5), (e) graph chart (C5)	39
4.1	Running Example	46
5.1	The core elements of the GUI for Faceted Search and Analytics	60

5.2	Example 3: Exploring the results of an analytic query with faceted search . .	61
5.3	Data of our running example	66
5.4	(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,	67
5.5	(a): Property-based transition markers, (b): Property Path-based transitions markers	69
6.1	The architecture of the system <code>RDF-ANALYTICS</code>	77
6.2	GUI of the system <code>RDF-ANALYTICS</code> for formulating the query "Average, sum and max price of laptops that have 2 to 4 USB ports, group by manufacturer and the origin of manufacturer"	78
6.3	(a) Tabular visualization of the results and (b) Loading of the results as a new dataset	78
6.4	2D and 3D visualization of results	79
6.5	3D visualization of results	80
7.1	Correspondence with OPAP operations	88
7.2	An example of roll-up and drill-down	89
8.1	Task-based evaluation with users: task completion and user rating	93
8.2	Task-based evaluation with users: total task completion and total user rating	94
8.3	An alternative implementation of the proposed model	96

List of Tables

3.1	Overview of Approaches and Systems for Category C1 (analytical queries A)	25
3.2	Overview of Approaches and Systems for Category C2 (analytical queries A)	29
3.3	Overview of the introduced Works of Category C4 (supporting mainly analytical queries of Category B)	35
3.4	Overview of the introduced Works of Category C5 (supporting analytical queries of Category B)	36
3.5	Comparing the functionalities of related systems	40
4.1	Feature Creation Operators	45
5.1	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 <code>temp</code>	73
5.2	For SPARQL-only evaluation approach	73
6.1	Efficiency - peak hours	81
6.2	Efficiency - off-peak hours	82

Chapter 1

Introduction

1.1 General Objective

Numerous Knowledge Graphs, expressed in RDF (Resource Description Framework), play a crucial role in consolidating and integrating data from diverse sources. In the realm of data analytics, the complexity and scale of these graphs continue to grow, highlighting an increasing demand for tools that facilitate analysis. However, the current provision of analytic services and systems that are effective, efficient, and user-friendly remains a challenge.

This thesis aims to address this challenge by developing a comprehensive method that delves into the intricacies of RDF Knowledge Graph analysis. The proposed method is destined to achieve two primary objectives: (i) prioritizing accessibility: the method (and the corresponding system) empowers ordinary users to analyze complex RDF Knowledge Graphs without requiring specialized technical expertise, (ii) alleviating the burden on experts: recognizing the time-intensive nature of analyzing RDF Knowledge Graphs, the method incorporates advanced features to enable experts to navigate and analyze these structures more efficiently, ultimately enhancing their productivity. By leveraging sophisticated design principles, user-centric interfaces and innovative features, the method aims not only to simplify but to enhance the entire process of analyzing RDF Knowledge Graphs. Its features collectively contribute to an environment where users can delve into the analysis without being burdened by the technical intricacies associated with RDF data. By combining accessibility for ordinary users and efficiency for experts, the envisioned method aims at establishing a new standard for inclusive and productive analytical environments in the realm of data integration and analysis.

1.2 Motivation and Vision

To leverage large scale data for gaining new insights, a recent and very promising practice in various domains (e.g. environment, health, economy, culture, economics and others), that has been adopted by both academia and industry is to construct a Knowledge Graph

(KG) [49] that aggregates and integrates data from several datasets, as illustrated in Fig 1.1.

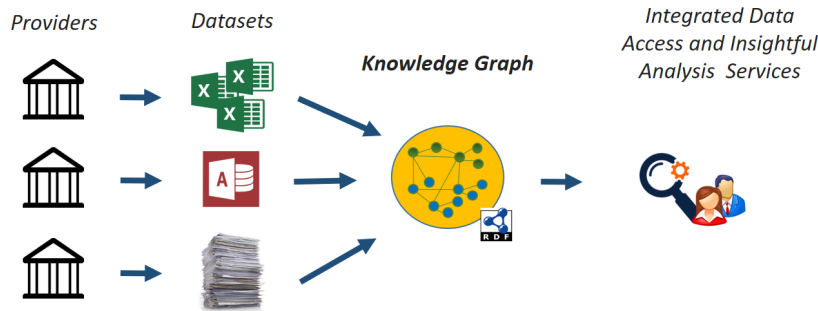


Figure 1.1: From Disparate and Fragmented Datasets to Knowledge Graphs

Several such graphs expressed in the W3C standard RDF (Resource Description Framework) exist, from *general purpose* KGs (like DBpedia [23] and Wikidata [120]) to *domain specific* semantic repositories, like Europeana [52] and [35] for the cultural domain, Drug-Bank [123] for drugs, GRSF [115] for the marine domain, WarSampo [59] for historical data, ORKG [54] for scholarly works, and [30, 91, 121] 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 [113]. Finally, Knowledge Graphs (KGs) are currently being employed for validating and enriching the responses generated by Large Language Models, such as ChatGPT (as in [78]).

Plain users can (i) *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. as in [25]), (ii) *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. as multi-perspective keyword search approach described in [81]), or (iii) use *interactive query formulators* that aim at aiding users in the formulation of structured queries (e.g. A-QuB [60], FedViz [31], SPARKLIS [39], and SPARQL-QBE [5]).

However, the analysis of big and complex KGs via *structured query formulation* is still very challenging for ordinary users and time-consuming as well as laborious for experts. This is due to their: (i) *complexity*: RDF KGs can be complex with large numbers of nodes and relationships. This can make it difficult for ordinary users to understand the structure of the graph and the relationships between the nodes, (ii) *lack of standardization*: there is no standard way to represent RDF knowledge graphs, which can make it difficult for users to interpret the data. This can lead to inconsistencies and errors in the analysis, (iii) *technical skills*: analyzing RDF knowledge graphs often requires technical skills such as programming and data analysis. Ordinary users may not have these skills, which can make it difficult for them to work with the data, (iv) *lack of tools*: there are a few user-friendly tools available for analyzing RDF knowledge graphs, which can make it difficult

for ordinary users to interact with the data. Most of the tools that exist are geared towards developers and require a high level of technical expertise. Furthermore, from a system perspective, efficiency is hard to achieve for very big KGs, while from an application/domain perspective users usually face completeness and freshness issues [122].

To emphasize these difficulties, let's consider a KG with information about products and its related entities (companies, persons, locations, etc.) with schema as shown in Fig. 1.2 (for reasons of brevity namespaces are not shown).

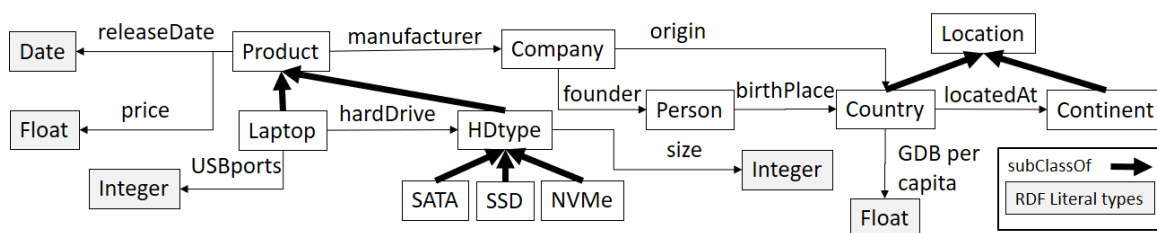


Figure 1.2: The schema of the running example

Suppose that we wanted to find “*the 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 information need would be expressed in SPARQL as shown in Fig. 1.3. Obviously, the formulation of such a query is quite difficult for ordinary users who do not have the required technical background knowledge.

For that, there is a need for an intuitive interactive model that will let any user regardless of expertise, to effortlessly formulate analytic queries without having any knowledge of the vocabulary (schema, ontology, thesauri) the actual contents of the dataset, or the syntax of the corresponding query language.

1.3 Research Question

This thesis centers around the following core research question: *Is it possible to come up with a general purpose method that enables interactive formulation of analytic queries over any RDF graph, that is appropriate for both ordinary and expert users?*

1.4 The Approach

We leverage the familiarity of users with *Faceted Search* [97] since this model allows expressing complex conditions through simple clicks. As illustrated in Fig. 2.1, our approach originates from a foundational model for faceted search over RDF data. Specifically, we build upon the core model for faceted exploration of RDF datasets, as introduced in [114]. To empower users with the capability to express both simple and complex queries seam-

```

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 >= "2021-01-01T00:00:00"^^xsd:dateTime &&
?rd <= "2021-12-31T00:00:00"^^xsd:dateTime)
} GROUP BY ?m

```

Figure 1.3: 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”.

lessly, we extend this model with interactive actions, facilitated by simple clicks. These actions correspond to a high-level query language for analytics, named HIFUN.

The expressive queries formulated in HIFUN are subsequently translated into SPARQL queries. These SPARQL queries are then executed against the relevant SPARQL endpoints to retrieve the desired results. To facilitate user understanding and interpretation, the obtained results are presented in the form of 2D and 3D charts. These visualizations serve to provide users with meaningful insights derived from the analytics conducted on the RDF data.

The proposed model is distinguished by the following characteristics: (i) applicability to any RDF dataset: it can be applied to any RDF dataset, independent of whether it follows a star-schema, (ii) query guidance: it provides guidance by supporting only answerable queries, ensuring that it never produces empty results due to lack of data, (iii) expressive power: it offers expressive power by supporting arbitrarily long paths in queries, (iv) count information: the model provides count information, enriching the analytical insights, (v) result restriction: it supports the interactive formulation of HAVING clauses for

restricting the final results, (vi) nested analytic queries: the model allows for the formulation of nested analytic queries, enabling users to explore data at various levels of complexity, and (vii) dual support: it accommodates both faceted search and analytic queries, offering flexibility in exploration techniques.

1.5 Contributions of this Dissertation

The key contributions of this thesis are the following:

(a) we propose a generic user interface for intuitively analyzing RDF Knowledge Graphs without pre-supposing any technical knowledge, (b) we extend the typical interfaces of faceted search systems with actions used for forming analytical queries of a high-level query language (called HIFUN) which is translatable to SPARQL, (c) we survey the approaches, systems and tools that enable the formulation of analytic queries over KGs expressed in RDF, (d) we define formally the state-space of the interaction model, we present the detailed steps of the algorithms used for producing the UI and we present an implementation of the model that showcases its feasibility, (e) we investigate the applicability of HIFUN over RDF KGs, we describe the data transformations that may be required and we show how the queries are translated from HIFUN to SPARQL, (f) we discuss about the expressive power of the proposed model, (g) we present an implementation of the model that showcases its feasibility, and (h) we report the results of a task-based evaluation with users which shows that the proposed model aid ordinary users in analyzing RDF KGs intuitive without having any technical background knowledge.

1.5.1 Publications produced in the context of this dissertation

The research activity related to this thesis has so far produced the following publications (ordered by publication date).

- [88] Papadaki, Maria-Evangelia, Yannis Tzitzikas, and Nicolas Spyros. “Analytics over RDF graphs.” *International Workshop on Information Search, Integration, and Personalization*. Springer, Cham, 2019.

This paper sketched an approach for applying analytics to RDF data, based on a high-level functional query language, called HIFUN. According to that language, each analytical query is considered as a well-formed expression of a functional algebra and its definition is independent of the nature and structure of the data. It also details the required data transformations, the translation of analytical queries from HIFUN to SPARQL and it introduces the primary implementation of a tool, developed for these purposes.

- [84] Papadaki, Maria-Evangelia, Nicolas Spyros, and Yannis Tzitzikas. “Towards

interactive analytics over RDF graphs.” Algorithms 14.2 (2021): 34.

This paper is an extension of [88]. It provides the *concrete algorithms* for translating analytical queries from HIFUN to SPARQL and it details the steps for developing a user-friendly interface for interactive analytics over any RDF graph. In particular, it describes a *unified logical interface* for letting users to (i) select the RDF file or triple store they want to analyze, (ii) specify and change the analysis context on the fly, (iii) formulate analytical queries using a high-level language for analytics (HIFUN), and (iv) shows how the queries will be translated from HIFUN to SPARQL.

- [116] Tzitzikas, Yannis, Maria-Evangelia Papadaki, and Manos Chatzakis. “A spiral-like method to place in the space (and interact with) too many values.” *Journal of Intelligent Information Systems* 58.3 (2022): 535-559.

This paper focuses on the problem of placing a set of values in the 2D (or 3D) space. It presents a novel family of algorithms that produces spiral-like layouts where the biggest values are placed in the center of the spiral and the smallest in the peripheral area, while respecting the relative sizes. The derived layout is suitable not only for the visualization of medium-sized collections of values, but also for collections of values whose sizes follow power-law distribution, since it makes evident the bigger values (and their relative size) and it does not leave empty spaces in the peripheral area which is occupied by the majority of the values which are small. The algorithm has linear time complexity (assuming the values are sorted), very limited main memory requirements, and produces drawings of bounded space, making it appropriate for interactive visualizations, and visual interfaces in general.

- [87] Papadaki, Maria-Evangelia, Yannis Tzitzikas, and Michalis Mountantonakis. “A Brief Survey of Methods for Analytics over RDF Knowledge Graphs.” *Analytics 2.1* (2023): 55-74.

This paper surveys the approaches, systems and tools that enable the formulation of analytic queries over KGs expressed in RDF. It identifies the main challenges, it distinguishes two main categories of analytic queries (domain specific and quality-related), and five kinds of approaches for analytics over RDF. Also, it describes in brief the works of each category and related aspects, like efficiency and visualization.

- [85] Papadaki, Maria-Evangelia, and Yannis Tzitzikas. “RDF-ANALYTICS: Interactive Analytics over RDF Knowledge Graphs.” *EBDT 2023*.

This paper introduces a novel system, called “RDF-ANALYTICS”, that enables ordinary users to formulate analytic queries over complex RDF knowledge graphs without requiring from them to have any background knowledge. It leverages the familiarity of users with Faceted Search (FS) systems and extends such a system with ac-

tions that let users formulate analytic queries, intuitively by exploiting a high-level language for analytics, called HIFUN, i.e. it builds upon [88], and demonstrates a particular GUI.

- [86] Papadaki, Maria-Evangelia, and Yannis Tzitzikas. “Unifying Faceted Search and Analytics over RDF Knowledge Graphs.” (2023) (Submitted to journal *Knowledge and Information Systems (KAIS)*, currently under revision).

This paper is an extended version of [85], it details all algorithms, it describes the expressive power of the model, it details the implementation, and includes the results of an evaluation with users.

Systems and Models

In the context of this thesis, the following systems and models were developed.

- (1a) <http://62.217.127.128:8080/3dvisualization/> is a system that visualizes the progress of COVID-19 virus over time by country in an interactive 3D environment. Adopting the metaphor of an urban area, it represents each country with a multi-storey cube. Each segment of the cube corresponds to a feature of the country and its volume is proportional to the value of that feature. The proposed system includes interactions and modification of the visualization parameters aiding users to explore the data, extract more details, and create new insights in an intuitive way.
- (1b) http://62.217.127.128:8080/3dvisualization_v2/ is an extension of the aforementioned system that lets users upload and visualize their own statistical data. The data is imported as a .csv file where the headers correspond to the attributes of analysis and the cells to the measure of it.
- (2) <http://demos.isl.ics.forth.gr/rdf-analytics/> is a system that enables plain users to formulate analytic queries over complex RDF knowledge graphs. It leverages the familiarity of users with faceted search and is extended with actions which are translated to a high-level query language for analytics, called HIFUN.
- (3) <http://62.217.127.128:8080/Interactive-RDF-Knowledge-Graphs-Analytics/> represents an enhanced and extended iteration of its predecessor (i.e. system (2)), focusing on improvements in user-friendliness and efficiency.

1.6 Outline of Dissertation

This thesis is organized as follows: Chapter 2 presents the background of this work. Chapter 3 describes the related work. Chapter 4 discusses about the applicability of HIFUN over RDE, the translation of HIFUN queries to SPARQL, and the algorithms used for the translation. Chapter 5 describes the proposed model. Chapter 6 delves into the model's implementation. Chapter 7 discusses about the expressing power of the model. Chapter 8

describes the evaluation process of the model. Finally, Chapter 9 concludes the paper and identifies issues for further research.

Chapter 2

Background

Here, we describe in brief the principles of (i) the Resource Description Framework (RDF) (in §2.1), (ii) the access methods over RDF data (in §2.2) (iii) analytical queries (in §2.3), (iv) the SPARQL Protocol and RDF Query Language (SPARQL) (in §2.4), (v) a high-level language for analytics, called HIFUN (in §2.5) and (vi) the Faceted Search (or Faceted Exploration) systems (in §2.6). A visual representation of their interconnections (and the general context) is given in Figure 2.1.

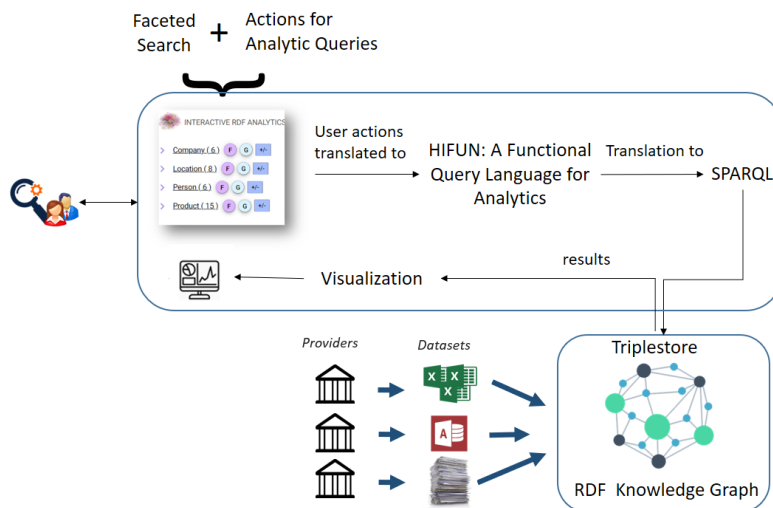


Figure 2.1: Context and main elements

2.1 The Resource Description Framework (RDF)

The Resource Description Framework (RDF) [12, 75] 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 etc.), the *predicate* to a characteristic of the entity (e.g. price of a product) and the *object* to the value of the predicate for

the specific subject (e.g. “1000”), as shown in Fig. 2.2. 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 dataset*).

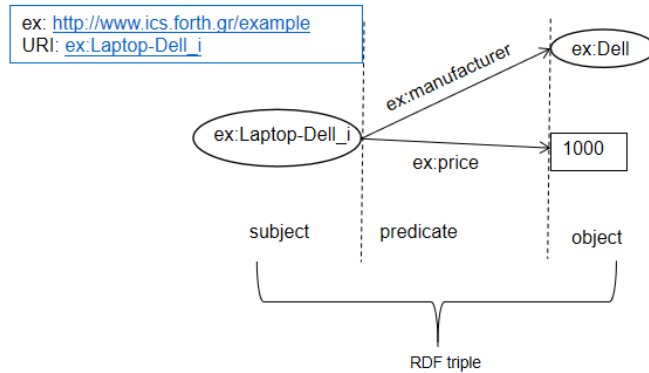


Figure 2.2: Example of an RDF triple

RDF Knowledge Graph A collection of RDF triples about related entities constructs an RDF knowledge graph that shows how these entities are related, as shown in Fig. 2.3.

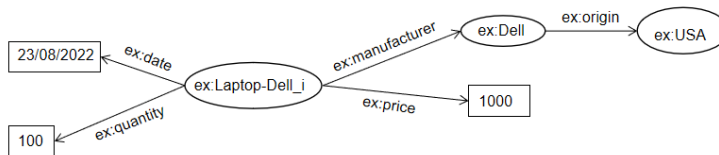


Figure 2.3: Example of an RDF graph

RDF Schema. RDF Schema¹ is a special vocabulary that enables the definition of schemas, that can be used for describing the resources in a more expressive, semantic, way. Its intention is to structure RDF resources, since even though RDF uses URIs to uniquely identify resources, it lacks semantic expressiveness. It uses classes to indicate where a resource belongs, and 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 $\langle \text{ex:Laptop rdf:type rdfs:Class} \rangle$ indicates that “Laptop” is a class, while the triple $\langle \text{ex:laptop-1 rdf:type ex:Laptop} \rangle$ indicates that the individual “laptop – 1” is an instance

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

of class *Laptop*. A property can be defined by using 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 $\langle \text{ex:manufacturer rdfs:type rdf:Property} \rangle$, $\langle \text{ex:manufacturer rdfs:domain ex:Product} \rangle$, $\langle \text{ex:manufacturer rdfs:range ex:Company} \rangle$, 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 $\langle \text{ex:manufacturer rdfs:subPropertyOf ex:producer} \rangle$ indicates that the property “manufacturer” is sub-property of “producer”. In addition, RDFS offers inference functionality² as additional information (i.e. discovery of new relationships between resources) about the data it receives. For example, if $\langle \text{ex:laptop-1 rdfs:type ex:Laptop} \rangle$ and $\langle \text{ex:Laptop rdfs:subclassOf ex:Product} \rangle$, then it can be deduced that “ $\text{ex:laptop-1 rdfs:type ex:Product}$ ”, i.e. that *ex:laptop-1* is also a Product.

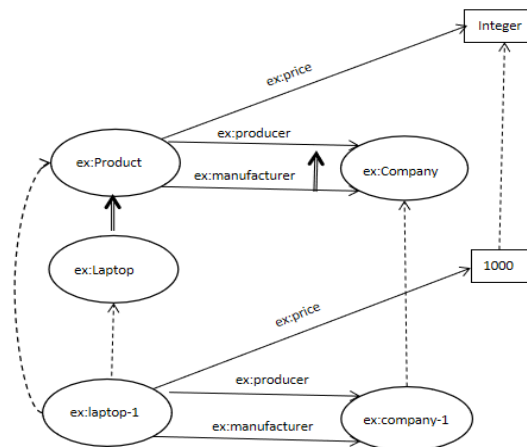


Figure 2.4: Example of an RDFS schema about products

2.2 The Spectrum of Access Methods over RDF

Access methods over RDF data are techniques used to retrieve and manipulate data stored in Resource Description Framework (RDF) format. There are several access methods that can be used to retrieve data from RDF stores, including:

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

Structured Query Languages. RDF data is mainly accessed using the SPARQL query language which is used to retrieve and manipulate data stored in Resource Description Framework (RDF) format. Apart from SPARQL, there are also a few other languages generally for KGs such as Cypher [42] (a declarative language implemented as part of the Neo4j graph database), Gremlin [9] (a combination of SQL, SPARQL and Cypher which focuses on navigational queries rather than matching patterns), PGQL [119] (an SQL-like pattern-matching query language), and G-CORE [10] (a graph query language that integrates the features provided by the graph query languages Cypher [42] and PGQL [119]) for querying property graphs.

Keyword Search systems. Such systems [81] allow users to search for documents, web pages, or other types of data by entering one or more keywords or search terms. The systems then retrieve and displays a list of documents or items that match the search criteria. While keyword search systems are useful for finding specific information quickly, they may also produce a large number of irrelevant results if the search terms are too general or if the system is not properly optimized.

Interactive Information Access. These methods aim to go beyond the traditional keyword search systems by allowing users to interact with the system and provide feedback on the relevance of the search results. Some common interactive information access methods include: (a) query reformulation: this method allows users to refine their search queries by suggesting alternative search terms or by providing a list of related concepts that can help users narrow down their search, (b) faceted Search: this method allows users to filter search results based on different criteria such as date, company, location etc., (c) recommender systems: these systems use machine learning algorithms to analyze a user's search history and provide personalized recommendations for content that might be of interest to the user, (d) collaborative filtering: this method uses data from multiple users to identify patterns and make recommendations based on the preferences of other users who have similar search histories or interests, (e) visual analytics: this method uses interactive visualizations to help users explore and analyze large datasets. It allows users to interact with the data and to discover patterns and insights that might not be immediately apparent from a traditional keyword search, and (f) Query-By-Example (QBE): this method enables users to refine their search queries interactively by providing examples or templates, allowing for a more intuitive and user-friendly approach to information retrieval (as in [5, 13, 28, 62].)

Natural language interfaces. These interfaces provide a more intuitive and accessible way for users to interact with computer systems using natural language, such as spoken or written language, as in [29]. Such interfaces use natural language processing (NLP) techniques to understand the meaning of user input and to generate appropriate responses.

Figure 2.5 illustrates the above methods and the distinctive characteristics of each one.

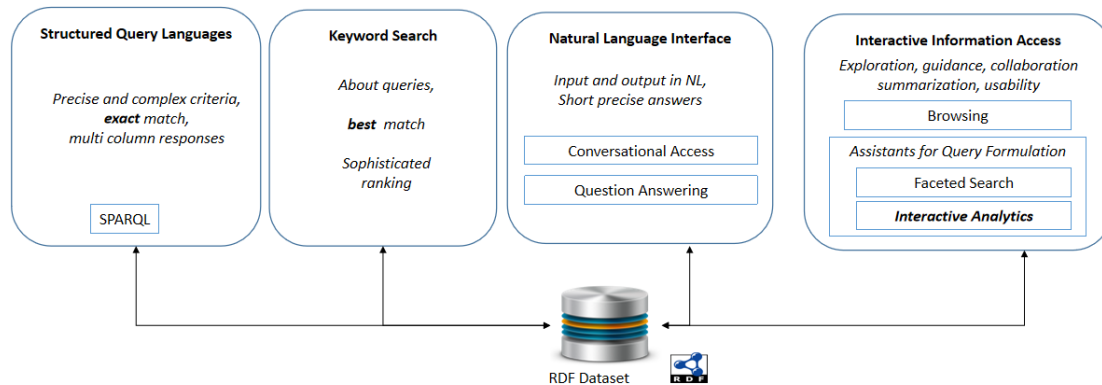


Figure 2.5: An Overview of the Access Methods over RDF

2.3 Analytical Query

An analytical query is a type of query designed to extract insights, patterns, or summaries from a dataset, often involving complex computations or aggregations. Analytical queries go beyond simple data retrieval and typically involve operations such as grouping, filtering, and statistical analysis to uncover meaningful information within the dataset. These queries are commonly used in data analysis and business intelligence to gain a deeper understanding of the underlying data.

An analytical query over RDF (Resource Description Framework) data extends this concept to the context of semantic data. It involves querying and analyzing this graph-based data model to extract patterns, relationships, and meaningful insights. In the context of RDF data, an analytical query may include operations such as aggregations over properties, exploration of relationships between entities, and statistical analysis to uncover patterns within the linked data. These queries enable users to gain a more comprehensive understanding of the semantic relationships and structure embedded in RDF datasets, making them valuable for knowledge discovery and exploration in various domains such as the Semantic Web and linked data environments.

2.4 SPARQL Protocol and RDF Query Language (SPARQL)

SPARQL³ is the standard query language used to retrieve and manipulate data stored in Resource Description Framework (RDF) format. It allows you to write queries that can be executed against RDF data stores, such as RDF databases or triple stores. These queries can be used to retrieve data, perform complex data manipulation, and perform advanced graph analysis. SPARQL includes a variety of features, including the ability to query for patterns, filter results, perform aggregation, and perform subqueries. From version 1.1,

³<https://www.w3.org/TR/rdf-sparql-query/>

SPARQL supports also complex querying using regular path expressions, grouping, aggregation, etc. In particular, and as regards analytic queries, SPARQL supports the modifier `GROUP BY` and supports various aggregate functions including `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`, and `GROUP_CONCAT`. In general, it is a powerful tool for working with RDF data and can be used in a variety of applications, including semantic web applications, data integration, and data analysis.

As an example, consider the KG of Figure 2.4 and suppose that we would like to express the query “total quantities of products released by company”. This query would be expressed in SPARQL as shown in Fig. 2.6.

```

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 (COUNT(?p) as ?total_products)
WHERE {
?p rdf:type ex:Product.
?p ex:manufacturer ?m.
} GROUP BY ?m

```

Figure 2.6: Expression in SPARQL of the query “total quantities of products released by company”.

However, even though SPARQL is a highly expressive query language (it supports a wide range of query operations, including pattern matching, filtering, aggregation, and subqueries), (i) it is a complex query language that requires significant expertise to use effectively (users must have a deep understanding of RDF data modeling, as well as the syntax and semantics of the SPARQL language itself), (ii) it can become extremely difficult to use (as RDF datasets grow in size and complexity, SPARQL queries can become increasingly difficult to manage and optimize), (iii) it may be necessary to use specialized query optimization techniques, such as query rewriting or materialization.

2.5 HIFUN: A Functional Query Language for Analytics

HIFUN [103] 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 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. For example, if the dataset D is a set of tweets, then the attribute “character count” (denoted as cc) is seen as a function $cc : D \rightarrow \text{Integers}$ such that, for each tweet t , $cc(t)$ is the number of characters in t . 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, as shown in Fig. 2.9. The attributes of a context are divided into two groups, the *direct* and the *derived*. The first group contains the attributes with origin D : these are the attributes whose values are given. The second group contains the attributes whose origins are different than D and whose values are computed based on the values of the direct attributes.

Consider a scenario where we have a dataset, D , comprising all delivery invoices from a distribution center (e.g., Walmart) over a year. These invoices provide details such as a unique identifier, delivery date, branch information, product type (e.g., CocaLight), and the quantity of units delivered for that product. Each type of product has its own dedicated invoice, and all the data for the year is stored in a database for analysis.

In a more abstract representation, the information in each invoice can be seen as a set of four functions: d for the date the product delivered, b for branch the delivery took place, p for the type of the product, and q for the quantity delivered for that type of product. This abstraction is illustrated in Fig. 2.8, where D denotes the set of all invoice numbers, and arrows indicate the attributes associated with each invoice.

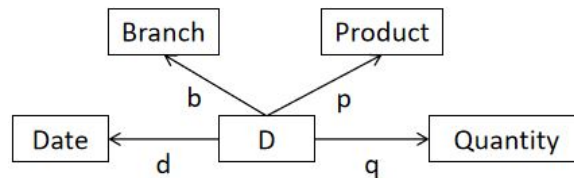


Figure 2.7: Representation of Invoices Dataset in HIFUN

Let’s say our objective is to determine the total quantity of products delivered to each branch over the entire year. For this computation, we only need information from two out of the four functions: b (branch) and q (quantity). A simplified dataset, as shown in Fig. 2.8 (a), illustrates the mappings returned by b and q for a set of seven invoices (numbered 1 to 7). To find the total quantity delivered to each branch, we can follow a three-step process:

Grouping. During the grouping step, we consolidate invoices that pertain to the same branch by utilizing the function b . This results in distinct groups of invoices for each branch, as illustrated below:

Branch b1: Invoices d1, d2

Branch b2: Invoices d3, d4

Branch b3: Invoices d5, d6, d7

Measuring. Next, in the measurement phase within each group, we identify the quan-

tity associated with each invoice, leveraging the function q :

Branch b1: Quantities 200, 100

Branch b2: Quantities 200, 400

Branch b3: Quantities 100, 400, 100

Reduction. Subsequently, in the reduction stage within each group, we aggregate the identified quantities:

Branch b1: Total quantity = $200 + 100 = 300$

Branch b2: Total quantity = $200 + 400 = 600$

Branch b3: Total quantity = $100 + 400 + 100 = 600$

The resulting association of each branch with its corresponding total quantity, as depicted in Fig. 2.8 (a), represents the desired outcome:

Branch-1 \rightarrow 300

Branch-2 \rightarrow 600

Branch-3 \rightarrow 600

The ordered triple $Q = (b, q, \text{sum})$ serves as a query over the dataset D , while the function $\text{ansQ} : \text{Branch} \rightarrow \text{TotQty}$ in Fig. 2.8 (b) stands as the answer to Q . The entire process outlined in Fig. 2.8 (a) constitutes the evaluation of the query. It's noteworthy that the association of branches with total quantities is made possible by the common source shared by b and q , which is D .

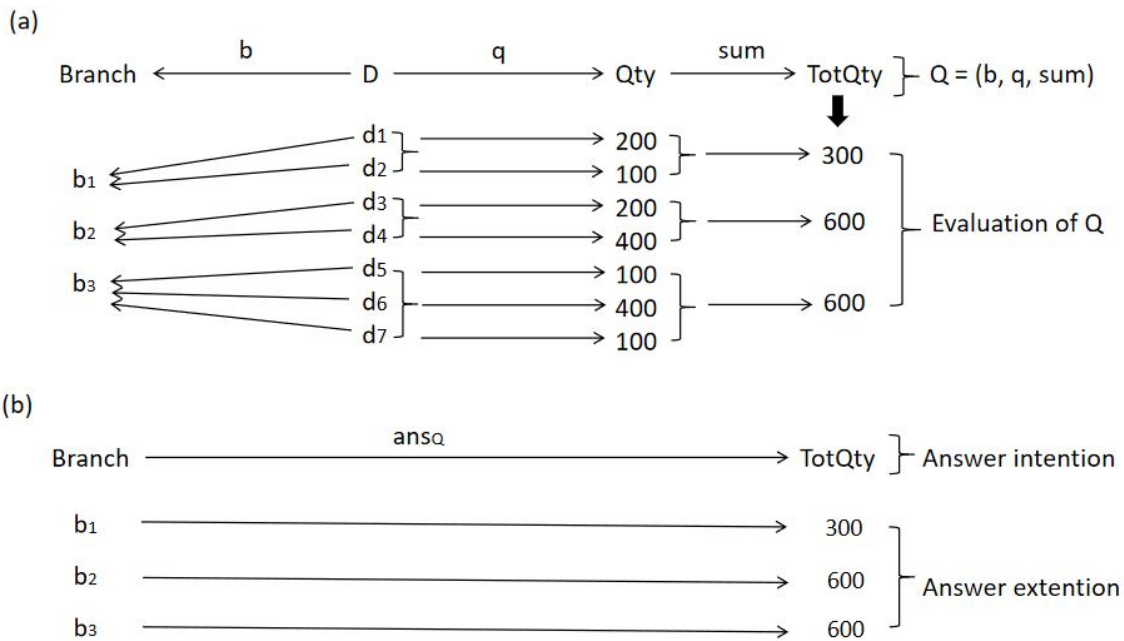


Figure 2.8: An analytic query and its answer

Consequently, 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* and it specifies how the data should be grouped or partitioned for analysis. The second component is called *measuring function* (or the *measure*) and it is the data value that is being analyzed or aggregated. The third component is called *aggregate operation* (or *reduction operation*) and it specifies the function used to summarize the data within each group or partition, such as sum, average, or count.

The grouping and the measuring function can also be more complex using the following four operations on functions: composition, pairing, restriction and Cartesian product projection. These operations form the so called functional algebra [102]. In addition, one can restrict the three components g, m, op of a HIFUN query. 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.

Roughly speaking, an analytical query Q is a path expression over an analysis context C ; a well formed expression whose operands are arrows from C and whose operators are those of the functional algebra. It is formulated using paths starting at the root and is evaluated in a three-step process, as follows: i) items with the same g -value g_i are grouped, ii) in each group of items created, the m -value of each item in the group is extracted from D and iii) the m -values obtained in each group are aggregated to obtain a single value v_i . Actually, the aggregate value v_i is the answer of Q on g_i . This means that a query is a triple of functions and its answer $AnsQ$ is a function, too.

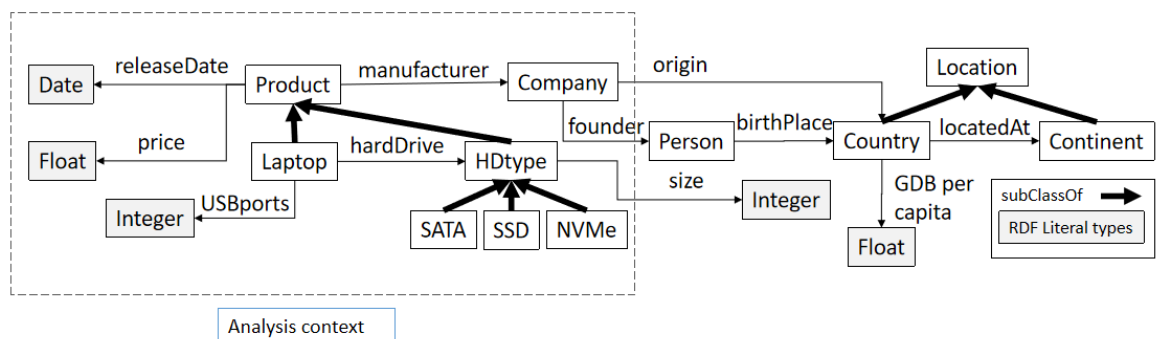


Figure 2.9: Example of an Analysis Context in HIFUN

2.5.1 Using HIFUN as an Interface to RDF Dataset

There are several ways in which HIFUN can be used, such as for studying rewriting of analytic queries in the abstract [103] or for defining an approach to data exploration [104]. In this work, we use HIFUN as a user-friendly interface for defining analytic queries over RDF datasets. To understand the proposed approach, consider a data source S with query language L (e.g. S could be a relational dataset and L the SQL language). In order to use HIFUN as a user interface for S , we need to (a) define an analysis context, that is a subset D of S to be analyzed and some attributes of D that are relevant for the analysis and (b) define a mapping of HIFUN queries to queries in L .

Defining a subset D of S can be done using a query of L and defining D to be its answer (i.e. D is defined as a view of S); and similarly, the attributes that are relevant to the analysis can be defined based on attributes of D already present in S . However, defining a mapping of HIFUN queries to queries in L might be a tedious task. In [104] such mappings have been defined from HIFUN queries to SQL queries and from HIFUN queries to MapReduce jobs.

The main objective of this work is to define a user-friendly interface allowing users to perform analysis of RDF datasets. To this end, we use the HIFUN language as the interface. In other words, we consider the case, where the dataset S mentioned above is a set of RDF triples and its language L is the SPARQL language.

The application of HIFUN over RDF will be detailed in Chapter 4.

2.6 Faceted Search

Faceted Search (or *Faceted Exploration*) [32, 90, 96] is a widely used interaction scheme for Exploratory Search. It is a type of search technique that allows users to filter and refine search results based on specific attributes or characteristics, known as facets as shown in Fig. 2.10. Facets are predefined categories or metadata that are associated with each search result, such as price range, product brand, released date, location, and so on. Faceted search typically presents users with a list of available facets and their corresponding values, which they can use to narrow down their search results. As the user selects or deselects different facets, the search results are dynamically updated to reflect their choices, as shown in Fig. 2.10. Faceted search is often used in e-commerce websites [95, 97, 112], online libraries [58, 107], and other digital collections that contain large amounts of heterogeneous data [14, 37, 68, 72, 114]. These systems can improve user experience by making it easier to find relevant information and reduce the amount of time spent searching through irrelevant results. 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.

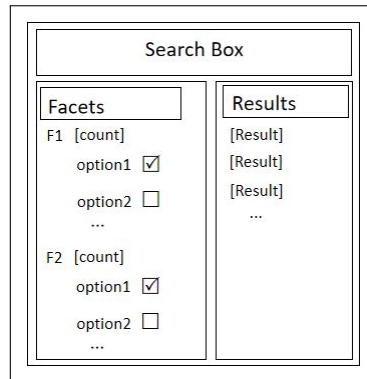
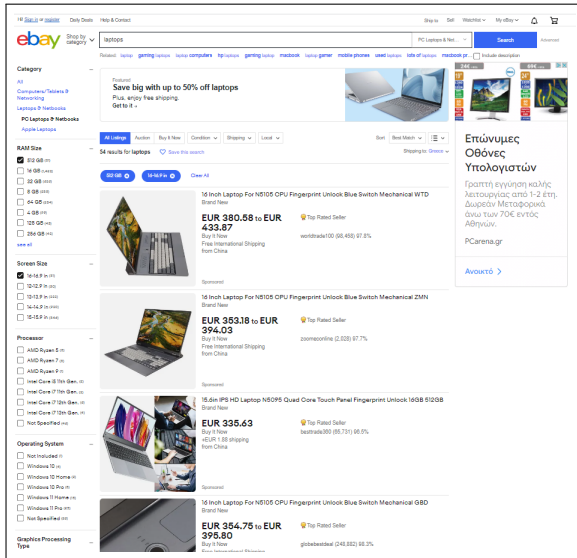
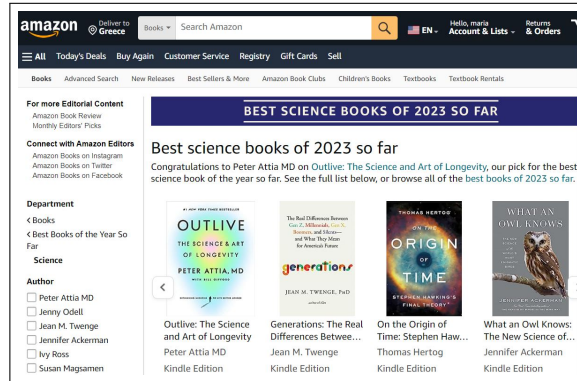


Figure 2.10: Faceted search system

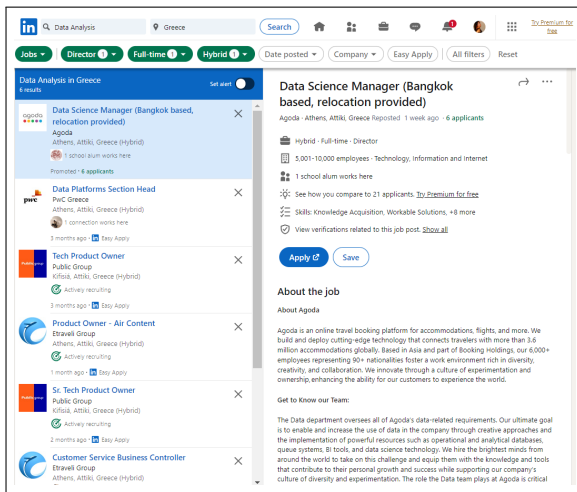
Below, we show some of the most popular Faceted Search systems. Fig. 2.11 (a) shows eBay, which is another major online marketplace, employs faceted search to enable users to narrow down their search results for products and auctions, offering filters like price range, condition, and seller location. Fig. 2.11 (b) shows Amazon.com which is a prominent e-commerce platform that utilizes faceted search to enhance user experience by allowing customers to refine their product searches based on various criteria such as price, brand, and customer ratings. Fig. 2.11 (c) shows LinkedIn Corporation that integrates faceted search into its professional networking platform, assisting users in finding job opportunities by refining searches based on factors like industry, location, and experience level. Finally, Fig. 2.11 (d) shows Booking.com, utilizes faceted search to streamline the process of hotel and travel reservations, allowing users to filter results based on criteria like price, location, and amenities. These platforms showcase the effectiveness of faceted search in various domains, providing users with tailored and efficient search experiences.



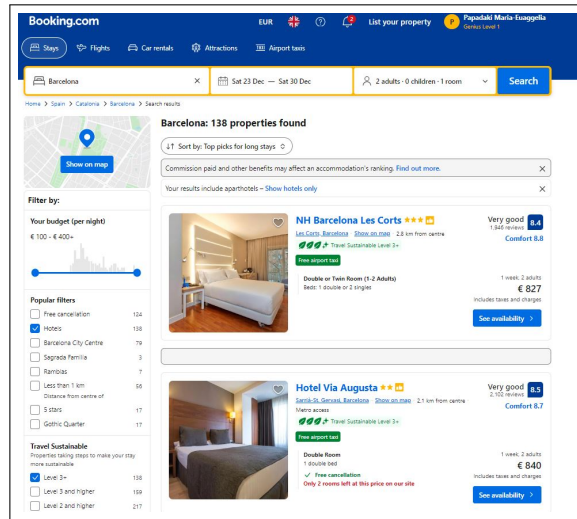
(a) eBay Inc.



(b) Amazon.com



(c) LinkedIn Corporation



(d) Booking.com

Figure 2.11: Examples of Faceted Search Systems

Chapter 3

Related Work

Section 3.1 describes in brief a few surveys about RDF Knowledge Graphs focused in integration, querying and visualization. Section 3.2 identifies the major challenges of RDF analytics. section 3.3 outlines the methodology we followed for finding the papers of this thesis, it reports their statistics and describes these works. Section ?? compares Faceted Search systems with a paradigm for interactive query formulation, named “Query-By-Example”. Section 3.4 discusses about the efficiency of the analytical approaches the surveyed papers propose and their ability to support visualization of the analytical results. Section 3.5 summarizes the related work. Finally, section 3.6 states the position and contribution of this thesis.

3.1 Past Surveys: Integration, Querying and Visualization of RDF Knowledge Graphs

There are several surveys related to RDF KGs, each addressing specific aspects of semantic data integration, querying techniques, and visualization approaches. For instance, [76] surveys approaches for large scale semantic integration of linked data emphasizing strategies for integrating multiple RDF datasets. [19] presents vocabularies dedicated to publishing RDF statistical data, while [8] surveys techniques and systems for querying RDF datasets focusing on the storage, the indexing and the query processing techniques for evaluating SPARQL queries. [118] introduces approaches for generating RDF graphs from heterogeneous data emphasizing mapping languages for schema and data transformations. Additionally, [2] categorizes OLAP approaches leveraging semantic web technologies according to several criteria including materialization, transformations and extensibility. Some surveys [11, 27] presents approaches for RDF KGs visualization and summarization for semantic RDF graphs [24].

These surveys play a crucial role in generating, integrating, querying, and visualizing RDF KGs, serving as prerequisite steps for producing analytics over RDF KGs. However, a significant gap in the literature exists - a survey specifically providing an overview of

analytics over RDF KGs, which is the core objective of our work.

3.2 RDF and Analytics: Challenges and General Approaches

In this section we identify the major challenges of analytics over RDF (Section 3.2.1), we provide a categorization of the existing works on RDF analytics (Section 3.2.2), and we present a classification for analytic queries by providing indicative examples (Section 3.2.3).

3.2.1 Challenges

A KG that integrates data from several datasets tends to have a complex structure in comparison to multi-dimensional data, since: (i) different resources may have different sets of properties (from different schemas), (ii) properties can be multi-valued (i.e. triples where the subject and predicate are the same but the objects are different) and (iii) resources may or may not have types. The typical methods for analytics that exist (i.e. over multidimensional data) are not adequate, since they presuppose a single homogeneous dataset, which in many cases requires prior transformations of the data. Furthermore, the semantics of RDF(S) are not always leveraged i.e. the inference based on `rdfs:subClassOf` and `rdfs:subPropertyOf`, and in many cases quality, completeness and freshness issues come across.

3.2.2 Categories of Works (related to RDF and Analytics)

We categorize the works related to RDF analytics in five basic categories, as illustrated in Figure 3.1. In particular, we classify them in: (C1) works that focus on the formulation of analytic queries directly over RDF (Section 3.3.2), (C2) works that first define data cubes over RDF data and then analyze it (Section 3.3.3), (C3) works that define domain-specific pipelines for producing RDF data and provide specific analytic services (Section 3.3.4), (C4) works that focus on publishing statistical data in RDF format (Section 3.3.5), and (C5) works that combine data from multiple sources for producing quality analytics (Section 3.3.6).

3.2.3 Categories of Analytic Queries

Here, we present the two main categories that analytic queries could be classified by providing some indicative examples:

- (A) *Domain specific analytic queries* related to the core information needs for which the KG was constructed, and they are expressible in SPARQL. These queries are mainly used in categories (C1)-(C3). Some indicative examples of such queries, from various domains expressed in natural language follow:

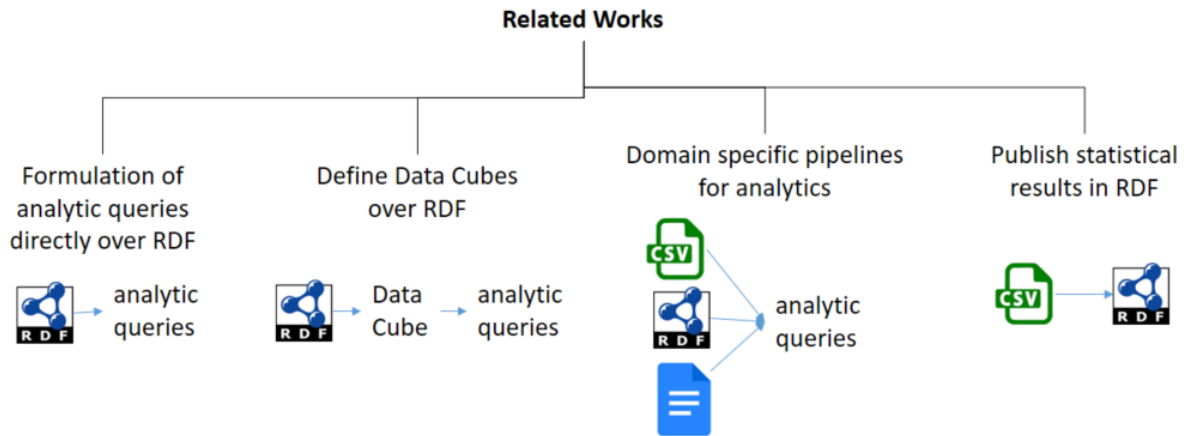


Figure 3.1: The spectrum of related works in 5 different categories

- *E-Commerce*: “average price of laptops made in 2022 from US companies that have 2 USB ports and an SSD drive manufactured in Asia grouped by manufacturer”
 - *Cultural domain*: “all paintings of El Greco grouped by exhibition country”
 - *Health/Covid*: “top countries with daily new covid19 cases per 1 million of population”
 - *Energy*: “energy spent at the University of Crete in the winter months group by hour”
 - *Transportation*: “average number of vehicles on Athens avenues during the morning peak hours (from 7 a.m. to 10 a.m.) in December of 2022”
 - *Education*: “average time spent on a task group by the frequency of participation in the course and the social status of the student”
 - *Sports*: “total goals and clean sheets of players of Spanish and England UEFA Champions League teams from 2021 to 2022”
- (B) *Quality-related analytics* (e.g., connectivity, data uniqueness, data verification) of one or more KGs, e.g., through statistics or specialized metrics. These queries are mainly used in categories (C4)-(C5). Some indicative examples of such queries are given below:
- Coverage of a dataset: “How many unique triples DBpedia offers for the entity Aristotle?”
 - Connectivity between Datasets: “Give me the number of common entities among DBpedia, Wikidata and National Library of France”

- Distribution of specific elements, such as properties, classes, namespaces, for detecting power-law cases in a KG or at the whole Linked Open Data (LOD) Cloud: “Is there a power-law distribution for the ontologies that are used from the LOD Cloud datasets?”
- Dataset Discovery: “Which dataset is the most relevant for the entity Socrates (e.g., offering the most triples)?”
- URI Quality: “What is the percentage of URIs that are dereferenceable and not broken?”

3.3 Survey of works and Systems

In this section, we detail the methodology we followed for finding papers that belong to the five basic categories of RDF analytics (C1-C5), we provide their statistics (Section 3.3.1), and we describe them in more details (Sections 3.3.2-3.3.6).

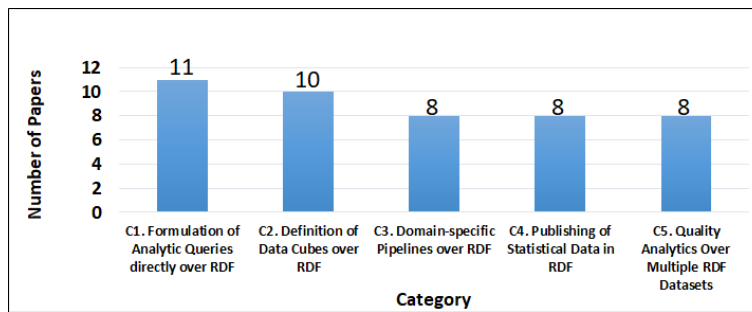


Figure 3.2: The number of surveyed works per category

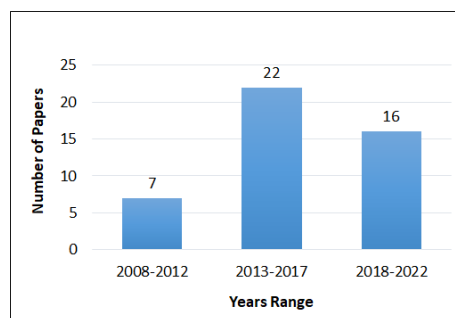


Figure 3.3: The publication year of the surveyed papers

3.3.1 Methodology and Statistics

For finding the related surveys we used Google Scholar in the period of June 2022 to November 2022 without any restriction on the publication date. Then, we typed on the search

Table 3.1: Overview of Approaches and Systems for Category C1 (analytical queries A)

Work/System	Evaluation	Offers	Visualiza- tion	Visualization Type	Year
Sridhar et al [105]	✓			-	2009
Ravindra et al [93]	✓			-	2010
Bikakis et al [22]		✓		Treemap, bar chart	2014
Zou et al [126]	✓			-	2014
Ibragimov et al [50]	✓			-	2015
Ibragimov et al [51]	✓			-	2016
Sherkhonov et al [100]				-	2017
Abdelaziz et al [1]	✓			-	2017
Ge et al [43]	✓			-	2021
FerrG• et al [41]	✓	✓		Table, map	2021
Papadaki et al [84]				-	2021

bar the following queries: (i) “RDF analytics tool”, (ii) “Interactive RDF analytics”, (iii) “RDF Data cube analytics”, (iv) “Efficiency of RDF data analytics”, (v) “Knowledge graph analytics” and (vi) “LOD Cloud analytics”. For each query, we analyzed manually the papers (from the first pages of Google Scholar results), i.e, by checking their title, abstract and body. We also found relevant papers from past surveys, e.g., for analytics over multiple datasets belong to the LOD Cloud [76]. Figure 3.2 shows some statistics about the number of the selected papers that fall in each category (C1-C5) and Figure 3.3 illustrates the publication year of these papers. We can see that the majority of the works we surveyed relates to the categories C1 and C2 and that most of the surveyed papers have been published between 2013-2017 (i.e., the most common case for the two mentioned categories). In addition, we can see that the most recent papers (i.e., between 2018-2022) relate mainly to domain-specific pipelines (i.e., category C3) and to approaches over multiple datasets at LOD scale (i.e., category C5).

3.3.2 C1. Formulation of Analytic Queries directly over RDF

Table 3.1 lists in chronological order some approaches relate to the formulation of analytic queries directly over RDE.

- [105] presents a language, called RAPID, for efficient expression of complex ana-

lytical queries over RDF data. The approach is based on integrating RDF-sensitive and advanced analytical query operators for analytical processing, called MD-join (which decouples the grouping and the aggregation clauses in query expressions) into Map-Reduce frameworks.

- [93] focuses on RDF data that includes several chain and star patterns. In particular, all patterns in the latter category can be processed concurrently using grouping-based operators for minimizing the I/O costs by computing sequentially the individual star patterns.
- [22] introduces “SynopsViz”, a Web-based visualization tool for scalable multi-level charting and visual exploration of very large RDF and Linked datasets. It performs a hierarchical aggregation, it incrementally retrieves data and generates visualizations based on user interaction. It provides statistical information of the dataset (e.g., number of triples, blank nodes, classes, subclasses, etc.) but this information is usually listed in tabular format leaving their interpretation to analysts. It also obtains computed statistics about the data being queried, such as: mean, variance, minimum and maximum values, etc. However, aggregate functions such as SUM and AVG are not supported. In the end, even though it is specialized in gathering statistical data about the dataset, it is not meant for traversing the dataset. Information displayed are not single resources, but a series of aggregated measures calculated over them. In addition, there is no evaluation report of this tool.
- [126] proposes some techniques to handle SPARQL queries with aggregate operators over dynamic RDF datasets, efficiently. It stores RDF data as a large graph and represents a SPARQL query as a query graph. To achieve efficient and scalable query processing, 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.
- [50] proposes a set of query processing strategies for executing aggregate SPARQL queries over federations of SPARQL endpoints by materializing the intermediate results of queries. However, the sources that participate in a federation might be unavailable at some point. The data and the schema of the sources might have evolved, since the federation was created; thus, integration rules might no longer be valid or history of the data will be lost.
- [51] shows how to process aggregate queries by using materialized views-named queries whose results are stored in a system (since they are typically much smaller in size than the original data and can be processed faster). These results are then used for answering subsequent analytical queries.

- [100] describes a possible extension of SemFacet [56] to support numeric value ranges and aggregation. The focus is on theoretical query management aspects, related to faceted search, however, it lacks an interface and implementation. From the mockups of the GUI, it seems that no count information is provided, whereas explicit path expansion is not supported. On the contrary, the authors use the notion of "recursion" to capture reachability-based facet restrictions. Since this approach is not implemented, no evaluation results are available.
- [1] presents Spartex, a vertex-centric framework for complex RDF analytics, that extends SPARQL to combine generic graph algorithms (e.g., PageRank, Shortest Paths, etc.) with SPARQL queries. It employs graph exploration and uses inter-vertex message passing during the query evaluation.
- [43] mentions that the existing federated RDF systems support only basic queries in SPARQL 1.0, and cannot be compatible with complex queries in SPARQL 1.1 well, such as aggregate queries. For this reason, proposes a query decomposition optimization method, which allows to combine triple patterns with the same multi-sources into one subquery. The schema can reduce the number of remote requests to improve the query efficiency by reducing the number of subqueries.
- [41] proposes an approach for guided query building that supports analytical queries in natural language and it can be applied over any RDF graph. The implementation is over the SPARKLIS editor [40] and it has been adopted in a national French project¹. During the query formulation no count information is provided, reducing in this way the exploratory characteristics of the process. The authors report positive evaluation results as regards the expressive power of the interactive formulator which works well on large datasets and is easier to use than writing SPARQL queries.
- [84] describes how a high-level functional query language, called HIFUN [103], can be exploited for applying analytics over RDF data. Rules for translating analytical HIFUN queries to SPARQL queries are presented. However, the interactive formulation of such queries and the evaluation part are missed from that work.

To the best of our knowledge, there is limited work regarding analytics directly over RDF graphs in a user-friendly and interactive environment. We managed to find only two such works [22, 41] that let users formulate analytical queries directly in such graphs by specifying the attributes of analysis (i.e. dimensions, measures) and the operations using drop-down menus or natural language and defining their values via checkboxes. The rest of the works [1, 43, 50, 51, 84, 93, 100, 105, 126] propose methods entangled with lower-

¹<http://data.persee.fr/explore/sparklis/?lang=en>

level technicalities, which prevent ordinary users from exploiting them while they are time-consuming and burdensome for experts.

Advantages. Analyzing directly an RDF graph (i) allows users to take full advantage of this rich data model, enabling them to explore and analyze data across multiple dimensions, (ii) allows users to leverage the semantics of RDF data improving the accuracy and relevance of their analysis, (iii) can easily intergate RDF data with other data sources, enabling users to combine and analyze data from multiple sources in a single view.

Disadvantages. Analyzing RDF data directly have some limitations: (i) can be complex, requiring a good understanding of the RDF data model and the SPARQL query language. This complexity can make it challenging for less experienced users to work with RDF data, (ii) can be computationally intensive, particularly for complex queries, (iii) there are currently no widely accepted standards for representing RDF data, which can make it difficult to work with data from multiple sources. This can result in data quality issues, such as inconsistencies or missing data, (iv) inaccurate or incomplete data can result in misleading or unreliable analysis, (v) there are currently relatively few tools available for working with RDF data directly, which can make it more difficult to analyze and visualize data.

3.3.3 C2. Definition of Data Cubes over RDF

To gap the mismatch between the relational data model and the graph data model there are approaches that define a data cube over existing RDF graphs and then apply OLAP. Table 3.2 lists such approaches in chronological order.

- [125] introduces Graph Cube to support OLAP queries effectively on large multi-dimensional networks. However, it usually ignores semantic information in heterogeneous networks. The experimental study shows that this tool supports decisions on large multi-dimensional networks, effectively.
- [48] introduces Linked Data Query Wizard, a web-based tool for displaying, accessing, filtering, exploring, and navigating Linked Data which are expressed in data cube format and stored in SPARQL endpoints. The main innovation of the interface is that it turns the graph structure of Linked Data into a tabular interface and provides easy-to-use interaction possibilities. It supports filtering of the columns (e.g., by a keyword or a numeric value), and simple aggregations. However, the tables are limited to the presentation of the direct neighborhood of entities (columns are entity properties, and column values are the objects of those properties) rather than results of arbitrary queries. Table cells can contain sets of values but not multi-column tables. The results of the conducted user study showed that the tool had a few weak spots that could be improved, but in general it is usable, both for experts and non-experts in computer science.

Table 3.2: Overview of Approaches and Systems for Category C2 (analytical queries A)

System/Work	Evaluation	Offers Visualization	Visualization type	Year
Zhao et al [125]	✓		-	2011
Hoefler et al [48]	✓	✓	Tabular	2013
Payola [57]	✓	✓	Various charts, i.e. line, bar, column, area, polar, pie, graph charts	2013
Vis-Wizard [111]	✓	✓	Various charts, e.g. bubble, pie, column, line, area, geo etc.	2014
Azirani et al [16]			-	2015
Jakobsen et al [53]	✓		-	2015
CubeViz [69]		✓	Various charts, e.g. pie, bar, column, line	2015
Benetallah et al [20]	✓		-	2016
Microsoft Power BI [38]		✓	Various charts e.g. bar, column, pie, area, treemap ect.	2016
Tableau ² [63]		✓	Various charts, e.g. column, bar, pie, line, area, map etc.	2019

- [57] presents Payola, a framework for Linked Data analysis and visualization. The goal is to provide end users with a tool enabling them to analyze Linked Data in a user-friendly way and without having knowledge about the SPARQL query language. This goal can be achieved by populating the framework with variety of domain-specific analysis and visualization plugins. Although, it encourages collaboration between users, e.g., experts can edit visualizations and SPARQL queries and lay-users can consume a result, it neglects to provide a complete representation of the dataset that is necessary for expressing the queries. At the same time, the amount of manual configuration and the necessary transformation steps between different abstractions might be considered a shortcoming by non-technical users. Regarding the evaluation of this tool, there is a concise report where the test users asked a couple of questions regarding usability of it and concludes that work on the usability is needed.
- [111] presents Vis-Wizard, a Web-based visualization system able to analyze multiple datasets using brushing and linking methods i.e. combining different visualizations to overcome the shortcomings of single techniques. The tool was designed for two different tasks: (i) explore endpoints like DBpedia and (ii) explore datasets that contain statistical data. Vis-Wizard allows users to group data and aggregate values providing multiple interactive widgets. According to [21] the online version reports a

multitude of errors, that prevented users to analyze the different visualizations that the tool offers. In fact, console errors rose and no charts appeared. Regarding endpoints like DBpedia, the tool works fine, but the tabular layout they implemented results to be a little messy at first. The evaluation conducted regarding the usability of the Vis-Wizard shows that while several usability issues still need to be fixed, the overall advantage is observable.

- [16] proposes algorithms that use the materialized result of an RDF analytical query to compute the answer to a subsequent query. The answer is computed based on the intermediate results of the original analytical query. However, the approach does not propose any algorithm for view selection. It is applicable for the subsequent queries and not to an arbitrary set of queries [51]. In addition, no evaluation is reported.
- [53] studies the improvement of SPARQL queries over QB4OLAP [34] (an extension the RDF Data Cube Vocabulary³ to fully support OLAP multi-dimensional models and operators) data cubes. The idea behind the proposed approach is to directly link facts (observations) with attribute values of related level members. Although preliminary results in an evaluation study show an improvement in queries performance, this approach prevents level members from being reused and referenced, breaking the Linked Data nature of QB4OLAP data instances.
- [69] proposes CubeViz, a user-friendly exploration and visualization platform for *statistical data* represented adhering to the RDF Data Cube vocabulary. If statistical data is provided adhering to the Data Cube vocabulary, CubeViz exhibits a faceted browsing widget allowing to interactively filter observations to be visualized in charts. However, it does not support aggregate functions, such as SUM, AVG, MIN and MAX, and blank nodes. Also, according to [3] if the created RDF Data Cube is sparse, it is possible to receive an empty result set after using the data selection component of CubeViz. As a consequence, CubeViz is not able to process all kinds of valid Data Cubes. In a domain-agnostic tool such as CubeViz, it is not feasible to integrate static mappings between data items and their graphical representations. Most of the chart APIs have a limited amount of pre-defined colors used for colouring dimension elements or select colors completely arbitrarily. Finally, this paper does not provide any information about the evaluation of this tool. It contains only a link to an online demonstrator letting users evaluate it.
- [20] presents multi-dimensional and multi-view graph data using MapReduce-based graph processing. The goal is to facilitate the analytics over the ER graph through summarizing the process graph and providing multiple views at different granularities. The technique, however, always materializes the result as paths with respect to a

³<https://www.w3.org/TR/vocab-data-cube/>

single entity identifier. The experiments conducted over real-world datasets, showed that the proposed approach performs well.

- [38] introduces Microsoft Power BI, a business intelligence platform that provides non-technical business users with tools for aggregating, analyzing, visualizing and sharing data. The interface of that tool is intuitive mainly for users who are familiar with Excel. It assumes that the ingested data has been cleaned up well in advance, while there is also a limit on its size (cannot import large datasets). After the data hit the limit, you have to upgrade to the paid version of Power BI. Also, the generated reports and dashboards can be shared only with those users who have the same email domains or the ones who have their email domains listed in your Office 365 tenant. At last, regarding the evaluation of that tool, there are provided comparative studies with other analytics tools as described in [94, 110].
- Tableau ⁴ [63] is a visualization tool capable of delivering interactive visualizations in no time, by using drag and drop. It offers a wide variety of options including pie, bar and bubble charts, maps, heat maps, scatter plots making use of which informative dashboards can be created instantly from diverse datasets. It performs aggregations, highlighting or drilling down in charts with much ease that even novice users can create visualizations to illuminate facts in a huge dataset. Tableau can be used to define and calculate new variables and perform simple data manipulations with usage of mathematical formulae like excel. However, initial data processing is needed which requires professional kit knowledge, while only column charts can be used for visualizing the results for free. Finally, no evaluation report is provided, however, there are available comparative studies with other tools [79, 92].

Overall, the aforementioned approaches follow common techniques in the formulation of the analytical queries. They let users specify the attributes of analysis (i.e. dimensions, measures) and the operations interactively using drop-down menus and define their values via check-boxes.

Advantages. Defining a Data Cube over RDF data have some advantages as: (i) enables users to explore data across multiple dimensions and thus summarizing and analyzing data at different levels of granularity, (ii) is designed for efficient querying, enabling users to quickly and easily retrieve data that are most relevant to their analysis and (iii) can integrate data from multiple sources, enabling users to analyze data from disparate systems or applications in a single view.

Disadvantages. Some of the disadvantages of defining a Data Cube over RDF data are: (i) defining a Data Cube may be complex to implement, requiring a good understanding of the RDF data model and the SPARQL query language, (ii) data must be transformed

⁴<https://www.tableau.com/>

into the RDF data model, which can be a time-consuming process. This may also be a barrier to adoption for organizations with large, complex datasets, (iii) there are currently relatively few tools available for working with RDF data cube data, which can make it more difficult to analyze and visualize data, and (iv) inaccurate or incomplete data may result in misleading or unreliable analysis.

3.3.4 C3. Domain-specific Pipelines over RDF

There are numerous works that focus on defining specific pipelines for constructing the desired KG from various structured and unstructured sources, and then offer particular analytic queries and visualizations to support domain-specific research purposes. Since there is a large number of such available cases (e.g., [4] surveys more than 140 papers on KGs from seven different domains), below we present a few number of indicative works from the medical, publications and cultural domain:

- *Medical Domain.* PhLeGrA [55] integrates data from several large scale biomedical datasets, for analyzing associations between drugs, i.e., for improving the accuracy of predictions of adverse drug reactions. [46] collects both structured and unstructured data for creating an aggregated KG about cancer data. The objective is to provide cancer data analytics through several services, such as treatment sequence analysis, data discrepancy analysis and others. [70] creates a KG, from over 50,000 articles related to corona viruses by using linked data techniques. The produced RDF dataset can be used for producing analytics through several extraction and visualization tools (e.g., it is feasible to analyze the number of articles that co-mention cancer types and viruses of the corona family). In addition, [98] describes a framework, called Knowledge4COVID-19, that integrates several RDF sources of COVID-19 related data. The resulting KG is exploited by machine learning methods for providing analytics and visualizations that are used for discovering adverse drug effects and for evaluating the effectiveness and toxicity of COVID-19 treatments.
- *Publications Domain.* OpenAIRE [67] is a Research KG that aggregates a collection of metadata and links, which are offered within the OpenAIRE Open Science infrastructure and provides several analytics and visualizations, such as for usage data (<https://usagecounts.openaire.eu/analytics>). Moreover, Open Research Knowledge Graph (ORKG) [15] exploits manual and automated techniques for creating and processing a scholarly KG. The mentioned KG can be used for further analysis through visualizations that are produced by the offered data science environments (e.g., see <https://orkg.org/visualizations>).
- *Cultural Domain.* FAST CAT [35] is a collaborative system for data entry and curation in Digital Humanities, and it can be exploited for performing historical analysis over

aggregated data. Moreover, [106] describes BiographySampo, an approach that provides analytics for biographical and prosopographical research by first transforming textual resources (from the National Biography of Finland) to RDF data. Afterwards, even users that are non-familiar with SPARQL can perform custom-made complex data analysis through the offered tools.

Overall, domain-specific pipelines can help to automate and streamline tasks such as data cleaning, transformation, integration, and analysis, which are specific to a particular domain.

Advantages. Domain-specific pipelines: (i) allow users to define a set of processing steps that extract, transform, and load RDF data into a more simplified form that make it easier for users to work with complex data, (ii) can be customized to meet the specific requirements of different users or applications, (iii) integrate data from multiple sources, enabling users to combine and analyze data from a variety of different domains or sources, (iv) can be automated, reducing the amount of manual effort required to process and analyze data improving efficiency and reducing the risk of errors, (v) can be reused across different projects or applications, enabling users to leverage existing pipelines and avoid duplicating effort.

Disadvantages. Defining domain-specific pipelines over RDF data has some limitations, as: (i) limited applicability since they are designed for specific domains or application areas, which means that they may not be easily transferable to other domains or applications. They require domain-specific knowledge and expertise to develop, which can limit their scalability and generalizability. (ii) complexity since developing such pipelines requires multiple steps such as data cleaning, transformation, integration, and analysis. This complexity can increase the risk of errors and require significant resources to implement. (iii) maintenance since once a domain-specific pipeline has been developed, it requires ongoing maintenance to ensure that it remains up-to-date and relevant. This can be a resource-intensive process, particularly if the pipeline relies on data from external sources that may change over time. (iv) interpretability since the results of a domain-specific pipeline can be difficult to interpret, particularly if the pipeline involves multiple steps or complex data transformations. This can make it challenging to identify errors or understand the reasoning behind the pipeline's outputs.

3.3.5 C4. Publishing of Statistical Data in RDF

This category relates to works for publishing statistical data. Publishing statistical data in RDF typically involves representing the data using standard vocabularies and ontologies that are designed for statistical data. Here, we present two vocabularies for publishing sta-

tistical data as linked data: (i) the *RDF data cube* vocabulary⁵, and (ii) the “Vocabulary of Interlinked Datasets” (*VOID* [6]). The related works are listed in Table 3.3 in chronological order.

- *Works with RDF data cube vocabulary.* [99] and [124] publish statistical data as linked data using the *RDF Data Cube* vocabulary. They use this vocabulary to define a set of classes and properties for describing the dimensions, the measures, the observations, and other elements of a multi-dimensional dataset.
- *Works with VOID vocabulary.* *VOID* is used for expressing metadata of one or more RDF datasets, i.e., for representing and publishing statistics like the number of triples, the properties or classes of the datasets or the number of links between different datasets. [65] presents “Aether” which generates, browses, measures and visualizes such statistics. [71] introduces “Loupe”, a tool that provides summaries and analyzes the vocabulary information of each RDF dataset (e.g., the classes and properties used in each dataset). [73] proposes possible extensions of *VOID* for publishing and analyzing connectivity analytics of semantic data warehouses, while other works like *SPORTAL* [47] and *SPLendid* [44] compute and publish such statistics for aiding the process of source selection of federated queries. Finally, [64] presents “KartoGraphI”, a tool for publishing statistical data for SPARQL endpoints through *VOID* (and extensions of *VOID*) and provides several kinds for visualizing the results.

Overall, void vocabularies provide a useful framework for organizing and describing datasets in a standardized way. They are also used for expressing and publishing statistics of RDF datasets providing thus valuable information about the structure, content, and quality of datasets. Void statistics enable users to analyze and understand the properties of datasets, which can help to identify potential errors, inconsistencies, or gaps in the data.

Advantages. Void vocabularies: (i) provide a standardized way to describe datasets, which improves interoperability (enable different systems to work together and exchange information seamlessly), (ii) provide metadata that helps to clarify the meaning and context of data, reducing ambiguity, (iii) include metadata that can be used to discover datasets, (iv) include information about the source and history of a dataset, which can be useful for tracking the provenance of data and ensuring its quality, (v) enable data to be reused more easily, by providing a standardized way to describe datasets, (vi) can be used to describe datasets of different sizes and complexities, from simple spreadsheets to large, distributed databases, (vii) can be extended with new terms and concepts as needed to describe specific types of data or domains.

Disadvantages. Void vocabularies (i) can be complex and difficult to understand for users who are not familiar with RDF or linked data, (ii) even though they provide a stan-

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

Table 3.3: Overview of the introduced Works of Category C4 (supporting mainly analytical queries of Category B)

Work/System	Analytical Queries for	Vocabulary	Publication Year
SPLENDID [44]	Statistics for SPARQL endpoints	<i>VoID</i>	2011
Salast et al [99]	Publication of Statistical data	<i>RDF data cube</i> vocabulary	2012
Zancanaro et al [124]	Publication of Statistical data	<i>RDF data cube</i> vocabulary	2013
Aether [65]	RDF Dataset Statistics	<i>VoID</i>	2014
VoIDWH [73]	Semantic Warehouse connectivity	<i>VoID</i> (+extensions)	2014
Loupe [71]	RDF Dataset Statistics	<i>VoID</i>	2016
SPORTAL [47]	Statistics for SPARQL endpoints	<i>VoID</i>	2016
KartoGraphI [64]	Statistics for SPARQL endpoints	<i>VoID</i> (+extensions)	2022

standard way to describe datasets, there is still some variation in how different organizations and communities use void. This can create inconsistencies in how datasets are described, making it more difficult to integrate data from different sources, (iii) are subject to change over time as new requirements emerge, and keeping up with these changes can require ongoing maintenance and updates to applications and tools that use void, (iv) is challenging to be used in all aspects of the data in a consistent and meaningful way as the datasets become larger and more complex.

3.3.6 C5. Quality Analytics Over Multiple RDF Datasets

This category relates to approaches that produce quality analytics over a single or multiple RDF datasets (even over LOD-Scale). As we can see in Table 3.4, most approaches of category C5 produce analytics either for measuring distributions (e.g., power-law cases) or for discovering datasets.

- *Works that measure distributions* (e.g., power-law). [108] measures and analyzes the graph features of Semantic Web (SW) schemas with focus on power-law degree distributions. The main finding of that work was that the majority of SW schemas (at 2008) having a significant number of properties (resp. classes) approximate a power-law for total-degree (resp. number of subsumed classes) distribution. [18] introduces “LOD-a-LOT”, an approach where 28 billion RDF triples from thousands of RDF documents was collected for enabling the analysis and the querying of combined data from multiple data sources, e.g., for analyzing the distribution of URIs

Table 3.4: Overview of the introduced Works of Category C5 (supporting analytical queries of Category B)

Work/System	Analytical Queries for	Based on	Number of Sources	Publication Year
Theoharis et al. [108]	Power-Law Distributions	Graph Metrics	250 RDF schemas	2008
LODVader [17]	Exploration, Dataset Discovery	Indexes	491 RDF datasets	2016
LODStats [33]	Coverage, Quality	Indexes	9,960 RDF datasets	2016
LOD-a-Lot [18]	Power-Law Distributions	Indexes	650K RDF documents	2017
LODsyndesis [75]	Connectivity, Dataset Discovery, Coverage	Indexes and Lattice-based measurements	400 RDF datasets	2018
Soulet et al. [101]	Elements Distribution	SPARQL Queries	114 RDF triplestores	2019
Haller et al [45]	Elements Distribution, Quality of URIs	SPARQL queries	430 RDF datasets	2020
LODChain [75]	Connectivity, Dataset Discovery	Real time lattice-based measurements	A single RDF dataset (connected at real time with 400 RDF datasets)	2022

and triples. [101] presents algorithms for computing analytical queries over Linked Open Data by aggregating the results of queries from running SPARQL endpoints (i.e., for producing analytics over multiple LOD datasets). This work measures the property and class usage on the LOD cloud, and they estimate the number of the available triples in the LOD Cloud. Finally, [45] presents an empirical analysis of linkage among all the datasets of the LOD cloud, by focusing on automated methods for analyzing different link types at scale. The objective was to analyze the availability and discoverability of LOD datasets, i.e., the most commonly used ontologies, namespaces and classes, and many others, e.g., for discovering power-law distributions, and to analyze the quality of URIs, e.g., broken links, deferenacable URIs, etc.

- *Works for Dataset Discovery.* LODVader [17] is a system that produces LOD analytics over 491 RDF datasets, for supporting dataset exploration, analysis and dataset

discovery. LODstats [33] is a service including some basic metadata and statistics for over 9,000 RDF datasets, e.g., for measuring the number of datasets of specific property and class elements. LODsyndesis [75] is a suite of services that provides analytics for measuring the *connectivity* among hundreds of RDF datasets. The target is the produced connectivity analytics to be exploited for improving the discoverability and reusability of the underlying datasets, and for answering coverage queries. Finally, LODChain [77] is a research prototype that computes *connectivity analytics* for a new RDF dataset at real time, to the rest of LOD Cloud through LODsyndesis, and produces several visualizations (including graph visualizations, bar and pie charts, etc.) and dataset discovery measurements. In particular, the target is the analytics to be used for enriching and verifying the content of the input dataset.

Overall, quality analytics over multiple RDF datasets involves the analysis and assessment of the quality of data across multiple RDF datasets. This process involves comparing and evaluating data from different sources to identify patterns and inconsistencies, and to determine whether the data meets a certain level of quality or not. The goal of quality analytics over multiple RDF datasets is to ensure that the data is accurate, reliable, and consistent, and that it can be used effectively for various purposes, such as data integration, analysis, and decision-making.

Advantages. Quality analytics over multiple RDF datasets, (i) can identify and correct errors, inconsistencies, or gaps in the data, improving decision-making and the overall quality of the data, (ii) enable users to combine and analyze data from different sources to gain new insights.

Disadvantages. Quality analytics over multiple RDF datasets (i) can be complex and difficult to implement and understand, especially for users who are not familiar with RDF or linked data, (ii) can be challenging, especially if the data is heterogeneous or has different formats or schemas, (iii) can be time-consuming and resource-intensive, especially if the data is distributed across multiple sources, (iv) requires ongoing effort and resources to ensure that the data is accurate and up-to-date. and (v) different datasets may use different vocabularies or ontologies, which can create interoperability issues and make it more difficult to integrate and analyze the data.

3.4 Efficiency and Visualization

This section discusses related aspects for the surveyed papers, i.e., efficiency (in Section 3.4.1) and visualization (in Section 3.4.2).



Figure 3.4: Indicative Screenshots of Visualization of analytical results for Categories C1-C2: (a) column chart (C1, C2), (b) bar chart (C2), (c) line chart (C2), (d) pie chart (C2), (e) bubble chart (C2), (f) geo chart (C2), (g) area (C2), (h) treemap (C1, C2), (i) graph (C2), (j) table (C2)

3.4.1 Efficiency

First, for the category C1, in [93], the authors measure the efficiency of joining star patterns with grouping operators for executing aggregating queries. They indicate that for complex analytical tasks that combine generic graph processing with SPARQL, vertex-centric graph processing frameworks are at least an order of magnitude faster than existing alternatives [1], whereas they demonstrate significant performance improvements for analytical processing of RDF data over existing Map-Reduce based techniques [105]. Also, they show that decomposing the analytical queries and materializing the intermediate results [50, 51] improve the query response time by more than an order of magnitude, and that in these cases the average query time increases linearly with the increase of dataset size [43].

Concerning the category C2, in [20] the authors show that the size of the dataset as well as the number of function operations in an analytical query influence the execution time of such a query. Also, they prove that running queries on Virtuoso over data cubes in the star pattern is faster than over cubes in the snowflake pattern, which is particularly interesting since the snowflake pattern is the pattern in which most RDF data cubes are available [53].

As regards category C3, in many cases the authors measure the execution time of the

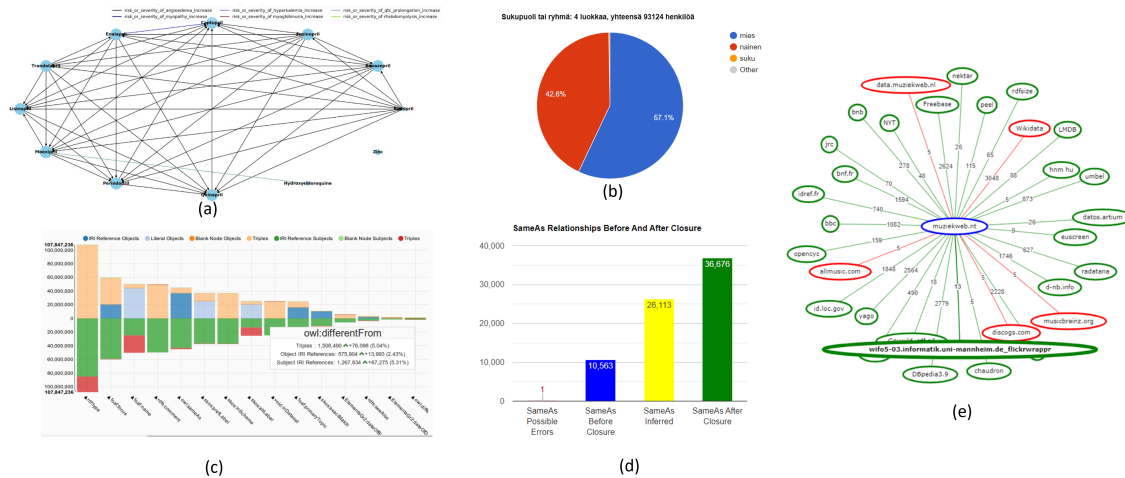


Figure 3.5: Indicative Screenshots of Visualization of analytical results for Categories C3-C5: (a) graph chart (C3), (b) pie chart (C3), (c,d) bar charts (C4,C5), (e) graph chart (C5)

SPARQL queries that produce the analytics [46, 98], which are executed over the resulting KG. Generally, these queries are executed quite fast, even in a few milliseconds. On the contrary, the most time consuming task of such domain specific approaches is usually the creation of the KG, which requires huge human effort [109].

Regarding the approaches of category C4, which produce statistics usually through SPARQL queries [33, 65], their performance highly depends on the underlying SPARQL endpoints, and the size of the datasets (number of triples, URIs, etc).

Concerning the category C5, for enabling the fast computation of analytics, in several cases specialized indexes are created, e.g., see LODsyndesis [75] and LOD-a-Lot [18]. Indicatively, the indexes of LODsyndesis aggregated KG [75] (which contain more than 2 billion triples), are constructed once in approximately 7 hours. On the contrary, the connectivity analytics are produced quite fast, i.e., even in a few seconds, by accessing the mentioned indexes. Regarding LODChain, it can produce the analytics for hundreds of thousands of triples in a few minutes (indicatively less than a minute for 50,000 triples), by also exploiting the indexes of LODsyndesis.

3.4.2 Visualization of Results

As regards the visualization of analytic queries results, most of the existing systems use popular types of charts. In particular, they use column charts, (e.g., [22, 38, 57, 69, 111]), bar charts, (e.g., [38, 57, 63, 65, 69, 77]), line charts (e.g., [57, 63, 69, 111]), pie charts (e.g., [38, 57, 63, 69, 106, 111]), bubble charts (e.g., [111]), geo charts (e.g., [63, 111]), area charts (e.g., [38, 57, 63, 111]), and graph charts (e.g., [57, 77, 98]). Finally, a few of them that support hierarchical data use treemaps (e.g., [22, 38]), while others follow more ordinary methods, i.e. tables (e.g., [41, 48]).

3.5 Summary

In summary, there are relatively few tools (managed to find two) currently available for working with RDF data directly in a user-friendly and interactive environment. Most of these tools require defining a Data Cube over RDF data before analyzing it, which demands a good understanding of the RDF data model and the SPARQL query language. Some of these tools define domain-specific pipelines that enable users to extract, transform, and load RDF data into a more simplified form, making it easier to work with complex data. However, these tools have limited applicability since they are designed for specific domains or application areas. The rest of the related work mainly focuses on expressing and publishing statistics of RDF datasets (using vocabularies like "Void") or analyzing the quality of data across multiple RDF datasets.

3.6 Our Positioning and Focus

In the realm of RDF Knowledge Graph analysis, we observe that are not so many works, neither running systems, that facilitate the easy and intuitive analysis of RDF Knowledge graphs. Our contribution falls within the domain of directly analyzing RDF Graphs (Section 3.3.2). We address existing gaps by introducing a user-friendly model that can be easily used by non-technical users. It will let users apply analytics effortlessly to any RDF data offering a familiar and gradual approach without requiring an in-depth understanding of dataset contents or the technical intricacies of SPARQL. Table 3.5 evaluates the most relevant systems, mentioned in Section 3.3.2, 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.

Table 3.5: 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
[100]	ANY	Yes	Yes	Yes but with No Count information	Not explicitly, reachability	No	No	No
[41]	ANY	Yes	No	No. Special interface	Not clear	No	Yes	Yes
[61]	ANY	Yes	No	Yes	Yes with counts	Yes	Yes	No
Our approach	ANY	Yes	Yes by AF	Yes	Yes with counts	Yes	Yes	Yes

Compared to existing systems, our approach distinguishes itself in several aspects: (i)

applicability: our approach can be applied to any RDF graph, eliminating the need for data pre-processing, unlike other systems limited to star-schema graphs, (ii) functionality: it not only supports analysis of the graph but also facilitates plain graph browsing, (iii) advanced features: the system supports result restrictions and property paths, enhancing its analytical capabilities, (iv) visualization: we provide visualization of analysis results, (v) user evaluation: our approach has undergone evaluation by users, validating its usability and effectiveness. Moreover, we emphasize its support for nested queries and user guidance, deriving from the inherent characteristics of Faceted Search.

Chapter 4

On Applying HIFUN over RDF

In this section, we discuss about the applicability of HIFUN over RDF (Section 4.1) and we describe the process of translating HIFUN queries to SPARQL queries (Section 4.2).

4.1 Applicability of HIFUN over RDF

Here, we investigate if RDF principles comply with the assumptions of HIFUN. We discuss about the prerequisites for applying HIFUN over RDF data (Section 4.1.1), and the possible data transformations that may be required (Section 4.1.2).

4.1.1 Prerequisites for Applying HIFUN over RDF Data

HIFUN can be applied over any dataset that has (i) uniquely identified data items and (ii) a set of attributes, where each attribute is seen as function that associates each data item with a value.

RDF data. The first assumption, the unique identification of the data items, is satisfied by the RDF data since each resource is identified by a distinct URI. The second assumption, the set of functional attributes, is partially satisfied by the RDF properties. The functional (i.e. `owl:FunctionalProperty`) or the effectively functional properties (i.e. even if they are not declared as functional, they are single-valued) satisfy this assumption since they do have only one value for each instance. However, there are cases in RDF where the properties may have (i) no or (ii) multiple values; a non-value property implies that a value may not exist (or it is unknown even if it exists) or it is incomplete, while a multi-valued property infers that the property has more than one values for the same resource. Hence, the existence of such properties require transforming the original data before applying HIFUN to it. Such transformations can be made by exploiting the feature operators or the facilities offered by the SPARQL language (Section §4.1.2).

RDF Schema. Each resource of an RDF schema is identified by a distinct URI; therefore, its data items are uniquely identified. However, a property (e.g. `rdf:type`, `rdfs:subClassOf` etc.) may appear more than once by relating different classes or classifying concepts in

more than one classes (i.e. a class might be sub-class of several super-classes). Nevertheless, these relationships are considered distinct, since they have different domain and/or range. Therefore, HIFUN supports analytics not only over RDF data, but over RDF schema(s), too.

Inference (i.e. discovering new relationships between resources) is supported, as well, since this process depends on the relations between the underlying data, independently if that data satisfies the assumptions of HIFUN.

4.1.2 Methods to Apply HIFUN over RDF

In this section, we describe how HIFUN can be applied over RDF data either (i) directly over the original RDF data, or (ii) after transforming the original RDF data.

Applying HIFUN over the Original RDF Data

Any class (i.e. set of resources) and any set of properties can be selected as the *root* and as the attributes respectively, of an analysis context. For example, any of the classes “Product”, “Laptop”, “HDType”, “Company”, “Person”, “Location”, “Country”, “Continent” of our running example Fig. 1.2 can be selected as the root of the context, and any of the properties “releaseDate”, “price”, “USBPorts”, “manufacturer”, “hardDrive”, “origin”, “founder”, “size”, “birthplace”, “locatedAt”, “GDBPerCapita” as the attributes of it.

Note that, any set of classes can be selected as the roots of a context as well, if the RDF graph consisted of more than one datasets. For example, both of the classes “Company”, “Product” of Fig. 1.2 can be selected as the roots of the context and any of the properties as the attributes of it.

Applying HIFUN over after Transforming the Original RDF Data

In case that RDF data does not satisfy the assumptions of HIFUN, a few feature operators (indicated in Table 4.1) for transforming it can be used. That table lists the nine most frequent *Linked Data-based Feature Creation Operators* (for short *FCOs*), as defined in [74], and they are re-grouped according to our requirements. f_i is a feature and $f_i(e)$ denotes the value of that feature for an entity $e \in E$. Each $f_i(e)$ is actually derived by the data that are related to e . \mathcal{T} denotes a set of triples, P a set of properties and p, p_1, p_2 properties. In detail,

- f_{co_1} suits to the normal case and it can be exploited to confirm that all the properties are functional e.g. the date that each product was delivered, the branch where each invoice took place. The value can be numerical or categorical.
- f_{co_2} and f_{co_3} relate to issues that concern missing and multi-valued properties and can be used for turning properties with empty values into integers.

- fc_{o4} can be used for converting a multi-valued property to a set of single-valued features, e.g. one boolean feature for each nationality, that a founder may have.
- fc_{o5} and fc_{o6} concern the degree of an entity and can be used to find the set of triples that contains a specific entity, defining its importance.
- fc_{o7} to fc_{o9} investigate paths in an RDF graph, e.g. whether at least one founder of a brand is “French”. It can be used for specifying a path (i.e. a sequence of properties p_1, p_2, \dots, p_n etc.) and treat it as an individual property p .

Table 4.1: Feature Creation Operators

id	Operator defining f_i	Type	$f_i(e)$
Plain selection of one property			
1	p.value	num/categ	$f_i(e) = \{v \mid (e, p, v) \in \mathcal{T}\}$
For missing values and multi-valued properties			
2	p.exists	boolean	$f_i(e) = 1$ if (e, p, o) or $(o, p, e) \in \mathcal{T}$, otherwise $f_i(e) = 0$
3	p.count	int	$f_i(e) = \{v \mid (e, p, v) \in \mathcal{T}\} $
For multi-valued properties			
4	p.values.AsFeatures	boolean	for each $v \in \{v \mid (e, p, v) \in \mathcal{T}\}$ we get the feature $f_{iv}(e) = 1$ if (e, p, v) or $(v, p, e) \in \mathcal{T}$, otherwise $f_{iv}(e) = 0$
General ones			
5	degree	double	$f_i(e) = \{(s, p, o) \in \mathcal{T} \mid s = e \text{ or } o = e\} $
6	average degree	double	$f_i(e) = \frac{ \text{triples}(C) }{ C }$ s.t. $C = \{c \mid (e, p, c) \in \mathcal{T}\}$ and $\text{triples}(C) = \{(s, p, o) \in \mathcal{T} \mid s \in C \text{ or } o \in C\}$
Indicative extensions for paths			
7	p1.p2.exists	boolean	$f_i(e) = 1$ if $\exists o2$ s.t. $\{(e, p1, o1), (o1, p2, o2)\} \subseteq \mathcal{T}$
8	p1.p2.count	int	$f_i(e) = \{o2 \mid (e, p1, o1), (o1, p2, o2) \in \mathcal{T}\} $
9	p1.p2.value.maxFreq	num/categ	$f_i(e) = \text{most frequent } o2 \text{ in } \{o2 \mid (e, p1, o1), (o1, p2, o2) \in \mathcal{T}\}$

These operators can be used for transforming the original RDF data via coding or by exploiting the SPARQL Query Language, i.e. CONSTRUCT queries¹ (used for deriving new RDF datasets) or nested queries.

Suppose that the pair (R, F) expresses an analysis context, where R is a set of resources and F the set of the features the objects in R have. Then, the resources R , as well as the features F could be defined by the triple patterns, i.e., “?s ?p ?o”, in the CONSTRUCT clause of a SPARQL query. Hereinafter, the bindings of “s” (or “o”) could correspond to the re-

¹<https://www.w3.org/TR/rdf-sparql-query/#construct>

sources, whereas the bindings of “*p*” to the set of features. In addition, nested queries² could also be used for defining these features of the analysis context.

In general any query translation method for virtual integration [66] can be employed. Please, note also that the list of feature operators can be expanded to cover any requirement that may arise.

4.2 Translation of HIFUN Queries to SPARQL

In this section, we detail the translation of HIFUN queries to SPARQL queries. Roughly, the grouping function yields variable(s) in the GROUP BY clause, the measuring function yields at least one variable in the WHERE clause, and the aggregate operation corresponds to an aggregate SPARQL function in the SELECT clause (over the measuring variable). The translation method is explained gradually below using examples based on the dataset of Figure 4.1. Please note that, for reasons of brevity we omit the namespaces).

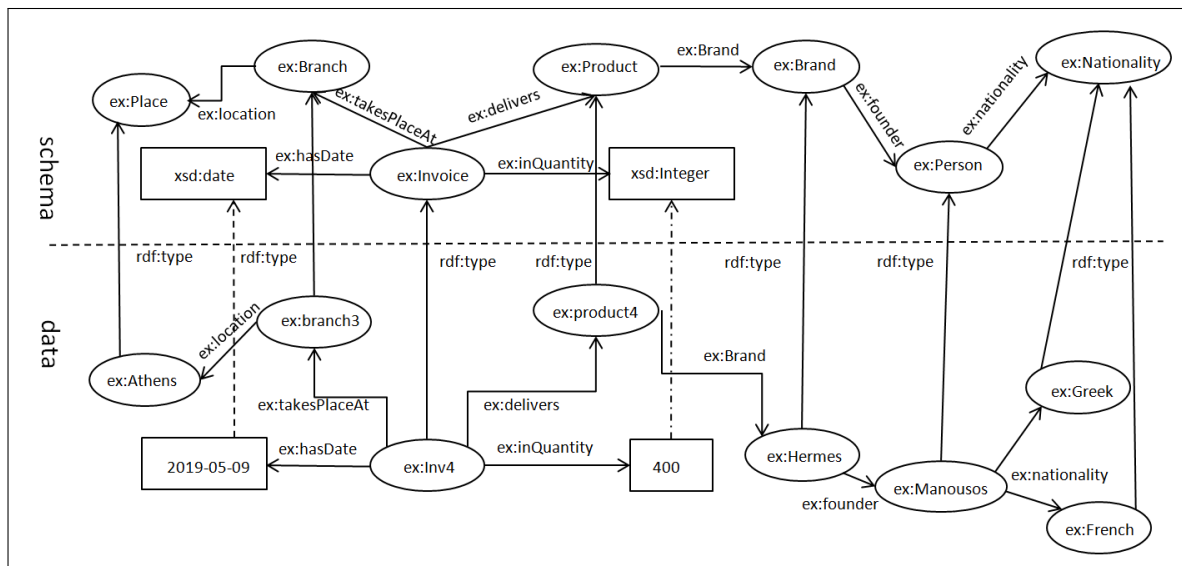


Figure 4.1: Running Example

4.2.1 Simple Queries

Suppose that, we would like to find the total quantities of products delivered to each branch. This query would be expressed in HIFUN as (*takesPlaceAt*, *inQuantity*, *SUM*) and in SPARQL as:

```
SELECT ?x2 SUM(?x3)
```

²<https://en.wikibooks.org/wiki/SPARQL/Subqueries>


```

WHERE {
  ?x1 ex:takesPlaceAt ?x2 .
  ?x1 ex:inQuantity ?x3 .
}
GROUP BY ?x2

```

Therefore, a HIFUN query (g, m, op) is translated to SPARQL as follows: the function g is translated to a triple pattern $?x1 \ g \ ?x2$ in the WHERE clause and the variable $?x2$ is added to the SELECT clause and in the GROUP BY clause. The function m is translated to a triple pattern $?x1 \ m \ ?x_N$ in the WHERE clause. The function op is translated to $op(right(m))$ in the SELECT clause, where $right(m)$ refers to the “right” variable of the triple pattern derived by the translation of m , i.e. $?x_N$, and op to the aggregate SPARQL function.

4.2.2 Attribute-Restricted Queries.

Suppose that, we would like to find the total quantities of products delivered to a particular branch, say “ $branch_1$ ”. This query would be expressed in HIFUN as $(takesPlaceAt/E, inQuantity, SUM)$, where $E = \{i \in D / takesPlaceAt(i) = branch_1\}$ and in SPARQL as:

```

SELECT ?x2 SUM(?x3)
WHERE {
  ?x1 ex:takesPlaceAt ?x2 .
  ?x1 ex:inQuantity ?x3 .
  ?x1 ex:takesPlaceAt branch_1 .
}
GROUP BY ?x2

```

Therefore a HIFUN query of the form $(g/v, m, op)$ is translated as follows: we translate the restriction v by adding in the WHERE clause the triple pattern $?x1 \ g \ v$. Please note that in this example, the restriction value refers to a URI, i.e. $branch_1$.

In case that the restriction referred to a literal value, then a FILTER statement “FILTER(?x1 op v)” would replace the triple pattern “?x1 g v” in the WHERE clause. For example, consider the following example, where the restriction is applied to the measuring function “total quantities of products delivered to each branch by considering only those invoices with quantity greater than or equal to 1”. This query would be expressed in HIFUN as $(takesPlaceAt, inQuantity/E, SUM)$, where $E = \{i \in D / inQuantity(i) \geq 1\}$ and in SPARQL as:

```

SELECT ?x2 SUM(?x3)
WHERE {
  ?x1 ex:takesPlaceAt ?x2 .
  ?x1 ex:inQuantity ?x3 .
  FILTER(?x3 ≥ xsd:integer("1")) .
}
GROUP BY ?x2

```

Consequently, a literal-attribute restriction in a HIFUN query $(g, m/cond, op)$ would be translated by adding in the WHERE clause the following constraint: $FILTER(right(m) cond)$.

4.2.3 Results-Restricted Queries.

Suppose that, we would like to find the total quantities of products delivered to each branch, but only for branches with total quantity greater than 1,000. This query would be expressed in HIFUN as, $(takesPlaceAt, inQuantity, SUM/F)$, where $F = \{g_i \in (g/E)/ans(g_i/E) \geq 1,000\}$ and in SPARQL as:

```

SELECT ?x2 SUM(?x3)
WHERE {
  ?x1 ex:takesPlaceAt ?x2 .
  ?x1 ex:inQuantity ?x3 .
}
GROUP BY ?x2
HAVING (SUM(?x3) > 1000)

```

Therefore we translate a result-restricted HIFUN query $(g, m, op/cond)$ by adding a HAVING clause with the following constraint: $HAVING Right(m) cond$ at the end of the query.

4.2.4 Complex Grouping Queries.

A grouping as well as a measuring function in HIFUN can be *more complex* when the following operations on functions as defined in [103] are used: *composition* (\circ) and *pairing* (\otimes). These operations form the so called functional algebra [102] and they are well known, elementary operations.

Composition

As we have already mentioned an attribute in HIFUN can be direct (i.e. attributes connect directly to the root of the analysis context) or derived (i.e. attributes computed from the values of direct attributes).

Direct attribute. Suppose that, we ask for the total quantities of products delivered by brand. This query would be expressed in HIFUN as $(brand \circ delivers, inQuantity, SUM)$, and in SPARQL as:

```
SELECT ?x3 SUM(?x4)
WHERE {
?x1 ex:delivers ?x2 .
?x2 ex:brand ?x3 .
?x1 ex:inQuantity ?x4 .
}
GROUP BY ?x3
```

Therefore, a HIFUN query $(f_k \circ \dots \circ f_2 \circ f_1, m, op)$, would be translated as follows:

For $k = 1$, the query would be considered as a simple query i.e. we would add the triple pattern $?x1 f_1 ?x2$ to the WHERE clause and the variable $right(f_1)$ to the SELECT and to the GROUP BY clause, as well.

For $k = 2$, i.e. composition of two functions $(f_2 \circ f_1)$, we would add the triple patterns $?x1 f_1 right(f_1)$ and $right(f_1) f_2 ?xf2r$ to the WHERE clause, and the variable $right(f_2)$ to the SELECT and to the GROUP BY clause.

For $k = n$, i.e. composition of n functions $(f_n \circ \dots \circ f_2 \circ f_1)$, we would add the triple patterns $?x1 f_1 right(f_1)$, $right(f_1) f_2 right(f_2)$, ..., $right(f_{n-1}) f_n right(f_n)$ to the WHERE clause, and the variable $right(f_n)$ to the SELECT and to the GROUP BY clause.

For $k = n + 1$, i.e. composition of $(n + 1)$ functions $(f_{n+1} \circ f_n \circ \dots \circ f_2 \circ f_1)$, we would add the triple pattern $right(f_n) f_{n+1} ?xnew$ to the WHERE clause, and replace the variable $right(f_n)$ with the $right(f_{n+1})$ in the SELECT and in the GROUP BY clause.

Derived attribute. Suppose that, we ask for the total quantities of products delivered by month. In this example, the attribute of “month” is considered a *derived* one. Such a query would be expressed in HIFUN as $(month \circ date, inQuantity, SUM)$ and in SPARQL as:

```
SELECT month(?x2) SUM(?x3)
WHERE {
?x1 ex:hasDate ?x2 .
?x1 ex:inQuantity ?x3 .
}
GROUP BY month(?x2)
```

Therefore, a HIFUN query of the form $(f \circ g, m, op)$, where f is an attribute derived from g , would be translated as follows: we would add to the WHERE clause the triple pattern $?x1 \ g \ ?xf2r$ (where $?xf2r$ is a brand new variable), and then, f would be derived from $right(g)$ by adding to the SELECT and to the GROUP BY clauses a SPARQL build-in function i.e. $f(right(g))$ (in our example $month(?x2)$); this function would extract the value f from that of $right(g)$.

Pairing.

Suppose that we would like to find the total quantities delivered by branch and product. This query would be expressed in HIFUN as $((takesPlaceAt \otimes delivers), inQuantity, SUM)$ and in SPARQL as:

```
SELECT ?x2 ?x4 SUM(?x3)
WHERE {
?x1 ex:takesPlaceAt ?x2 .
?x1 ex:inQuantity ?x3 .
?x1 ex:delivers ?x4.
}
```

GROUP BY ?x2 ?x4

Therefore, a HIFUN query $(f_k \otimes \dots, \otimes f_2 \otimes f_1, m, op)$ would be translated as follows: we would add the triple patterns $?x1 f_1 \ right(f_1), ?x1 f_2 \ right(f_2), \dots, ?x1 f_k \ right(f_k)$ to the WHERE clause and the variables $right(f_1), right(f_2), \dots, right(f_k)$ to the SELECT and to the GROUP BY clauses. In other words, we would join the pairing functions i.e. f_1, f_2, \dots, f_k on their shared variable i.e. $?x1$.

4.2.5 The Full Algorithm for Translating a HIFUN Query to a SPARQL Query

Let us now describe the full translation algorithm of HIFUN queries to SPARQL queries.

Let $q = (gE/rg, mE/rm, opE/ro)$ be a HIFUN query where

- gE is the grouping expression,
- mE is the measuring expression,
- opE is the operation expression, and
- rg is a restriction on the grouping expression,
- rm is a restriction on the measuring expression, and

- ro is a restriction on the operation expression.

This query is translated in SPARQL as follows:

```
Q = "SELECT " + retVars(gE) + " " + opE(mE) + "\n"
+ "WHERE {" + " \n"
+ triplePatterns(gE) + " \n"
+ triplePatterns(mE) + " \n"
+ "}" + " \n"
+ "GROUP BY " + retVars(gE) + " \n"
+ "HAVING " + restr(Qans)
```

In order to formulate such a query, we create and concatenate the strings it consists of.

1. We start with the translation of the grouping expression gE by creating a string for the triple patterns the terms g_i of gE participates, i.e. $triplePatterns(gE) += ?x_i g_i right(g_i)$. If gE contains any restriction rg we supplementarily create a string for the triple pattern which expresses that constraint:
 - 1.1. if rg refers to a URI, then $triplePatterns(gE) += ?x_i g_i rg$,
 - 1.2. if rg refers to a LITERAL, then $triplePatterns(gE) += FILTER(right(g_i) rg)$, (as described in §4.2.2).
2. We proceed with the translation of the measuring expression mE by creating a string for the triple patterns in which the terms m_i of mE participates, i.e. $triplePatterns(mE) += x_i m_i right(m_i)$. If mE contains any restriction rm we supplementarily create a string for the triple pattern which expresses that constraint:
 - 2.1. if rm refers to a URI, then $triplePatterns(mE) += ?x_i m_i rm$,
 - 2.2. if rm refers to a LITERAL, then $triplePatterns(mE) += FILTER(right(m_i) rm)$.
3. Then, we create a string for the returned variables, $retVars(gE) += right(g_i)$.
4. In the end, we translate the aggregate expression opE by creating a string for the aggregate operation op which apply over the measuring function mE , i.e., $opE(mE) = op(right(m_i))$.
5. Optionally, if any restrictions re are applied to the final answer Q_{ans} , then we create a string for expressing these restrictions, i.e. $restr(Q_{ans}) = right(m_i) re$.

Example. Suppose that we would like to find the total quantities by branch and brand only for the month of January, by considering only (a) the invoices with quantity greater than or equal to 2, and (b) the branches with total quantity greater than 1,000. This query would be expressed in HIFUN as

$(takesPlaceAt \otimes (brand \circ delivers)) /_{month=01}, inQuantity_{\geq 2}, SUM /_{>1,000}$.

Following the translation steps described before, the corresponding query in SPARQL would be:

```
SELECT ?x2 ?x5 SUM(?x3)
WHERE {
  ?x1 ex:takesPlaceAt ?x2 .
  ?x1 ex:inQuantity ?x3 .
  ?x1 ex:delivers ?x4 .
  ?x4 ex:brand ?x5 .
  ?x1 ex:hasDate ?x6 .
  FILTER((MONTH(?x6) = 01) && (?x3 >= xsd:integer("2")))
}
GROUP BY ?x2 ?x5
HAVING (SUM(?x3) > 1000)
```

Pseudo-code of the algorithm

Here, we provide the pseudocode describing the aforementioned steps for translating a HIFUN query to a SPARQL query.

Simple query: Algorithm 1 expresses the translation of queries with simple grouping and measuring functions that may involve restrictions too; i.e., queries of the form $Q = (g/rg, m/rm, op)$.

Composition: Algorithm 2- Composition describes the translation of queries that involve composition(s) in the grouping or the measuring function, e.g., queries of the form $(f_k \circ \dots \circ f_2 \circ f_1, m, op)$. An extension of this algorithm for supporting *derived* attributes, too, is given in Algorithm 3. Please, note that all predefined functions of SPARQL with one parameter can be used straightforwardly as derived attributes.

Pairing: Algorithm 2- Pairing describes the translation of queries that involve pairing(s) in their grouping or measuring function(s), e.g., queries of the form $(f_k \otimes \dots \otimes f_2 \otimes f_1, m, op)$.

Algorithm 1 Algorithm for computing the components of the translated query for the **Simple Case**

Require: A HIFUN query $q = (g/rg, m/rm, op/ro)$

Ensure: $retVars(g)$, $op(m)$, $triplePatterns(g)$, $triplePatterns(m)$, $retVars(g)$, and $restr(Q_{ans})$

```

1: right(g) ← newVariable()
2: triplePatterns(g).concat(?x1 g right(g))           ↗Grouping function
3: if  $rg \langle \rangle \epsilon$  then                             ↗if  $rg$  is not empty
4:   if  $rg.endsWithURI()$  then                       ↗Group restriction involving a URI
5:     triplePatterns(g).concat(?x1 g rg)             ↗The restriction is expressed as a triple pattern
6:   else                                             ↗Group restriction involving a literal
7:     triplePatterns(g).concat(FILTER(right(g) rg))  ↗The restriction is expressed as a filter
8: right(m) ← newVariable()
9: triplePatterns(m).concat(?x1 m right(m))          ↗Measuring function
10: if  $rm \langle \rangle \epsilon$  then                            ↗if  $rm$  is not empty
11:   if  $rm.containsURI()$  then                       ↗Measuring restriction involving a URI
12:     triplePatterns(m).concat(right(m) m rm)       ↗The restriction is expressed as a triple pattern
13:   else                                             ↗Measuring restriction involving a literal
14:     triplePatterns(m).concat(FILTER(right(m) rm))  ↗The restriction is expressed as a filter
15: retVars(g).concat(right(g))                       ↗for the SELECT and the GROUP BY clauses
16:  $op(m) = op(right(m))$                                ↗for the SELECT clause
17:  $restr(Q_{ans}) = right(m) ro$                        ↗for the HAVING clause

```

Pairing of compositions: Algorithm 2- PairingAndComposition describes the translation of queries that involve pairing(s) of composition(s), i.e., queries of the form $(gc_1 \otimes \dots \otimes gc_k, m, op)$, where each gc_i is an individual function or a composition of functions. Such expressions are translated as follows: $translate(gE) = translatePairing(translateComposition(gc_1) \otimes \dots \otimes translateComposition(gc_k))$.

General case: Algorithm 4 describes the translation of queries in the general case, where composition(s) occur in the restrictions of the grouping or the measuring function(s), e.g., queries of the form $Q = (g/gc_1 \circ gc_2 \circ \dots \circ gc_n = rg, m/mc_1 \circ mc_2 \circ \dots \circ mc_n = rm, op)$, where rg and rm are not necessarily single URIs or literals, but path expressions that end with a URI or literal.

4.2.6 Cases where the Prerequisites of HIFUN are not Satisfied

In order to apply HIFUN over a dataset, it should consist of functional attributes. However, in RDF data, (a) incomplete information or (b) multi-valued properties may exist.

Incomplete Information. Certain properties or attributes of a resource may have missing or unknown values. For example, consider that, the schema of Fig. 4.1 contained the

Algorithm 2 Auxiliary algorithms for compositions and pairings

```

1: procedure COMPOSITION( $f_c = f_k \circ \dots \circ f_1$ )    ↗returns the triplePatterns and the retVars for a
   composition
2:    $tp \leftarrow ""$ ;
3:   for  $i \in 1..k$  do
4:      $right(f_i) \leftarrow newVariable()$ 
5:     if  $k=1$  then
6:        $tp.concat(?x1 f1 right(f_1))$ 
7:     else
8:        $tp.concat(right(f_{i-1}) fi right(f_i))$ 
9:   return  $tp, right(f_k)$ 

1: procedure PAIRING( $f_p = f_1 \otimes \dots \otimes f_k$ ) ↗returns the triplePatterns and the retVars for a pairing
   expression
2:    $tp \leftarrow ""$ ;  $retVars \leftarrow ""$ 
3:   for  $i \in 1..k$  do
4:      $right(f_i) \leftarrow newVariable()$ 
5:      $tp.concat(?x1 fi right(f_i))$ 
6:      $retVars.concat(right(f_i))$ 
7:   return  $tp, retVars$ 

1: procedure PAIRINGOVERCOMPOSITIONS( $f_p = g_{c_1} \otimes \dots \otimes g_{c_k}$ )    ↗Pairing over Compositions
2:    $tp \leftarrow ""$ ;  $retVars \leftarrow ""$ 
3:   for  $i \in 1..k$  do
4:      $tp.concat(Composition(g_{c_i}).tp)$ 
5:      $retVars.concat(Composition(g_{c_i}).retVars)$ 
6:   return  $tp, retVars$ 

```

property “ex:birthYear”, with domain the class “ex:Person” and range an integer-typed literal. In that case, the problem of (a) *incomplete information* arises, i.e., since the dataset may not contain information about all the birth year of all the persons (for a variety of reasons, including missing data, privacy concerns, or simply because the information is not known). However, handling incomplete information is an important consideration when working with RDF data, as it can affect the accuracy and completeness of any analysis or processing performed on the data. For that, we could exploit the feature operator FCO_1 of Table 4.1; a feature that confirms that all the properties are functional. In particular, for each triple $\langle s, p, o \rangle$, if the value of p is null i.e., $o = \emptyset$, we replace it with 0 i.e., $\langle s, p, 0 \rangle$, otherwise, we replace it with 1, i.e., $\langle s, p, 1 \rangle$.

Algorithm 3 Algorithm for composition if derived attributes are involved

```

1: procedure COMPOSITIONSUPPORTINGDERIVED( $f_c = f_k \circ \dots \circ f_1$ )
2:    $tp \leftarrow ""$ ;  $retVars \leftarrow ""$ ;
3:   for  $i \in 1..k$  do
4:     if  $f_i$  is not a derived attribute then
5:        $right(f_i) \leftarrow newVariable()$ 
6:       if  $i=1$  then
7:          $tp.concat(?x1 \text{ fi } right(f_i))$ 
8:       else
9:          $tp.concat(right(f_{i-1}) \text{ fi } right(f_i))$ 
10:       $retVars \leftarrow right(f_i)$ 
11:     else ▷Is a derived attribute
12:       $retVars \leftarrow f_i(retVars)$  ▷No triple pattern will be produced
13:   return  $tp, retVars$ 

```

Multi-valued properties. In RDF, a resource can have multiple instances of a particular property, each with a different value. For example, consider that a brand has been founded by more than one persons. That means that the “founder” property is multi-valued, allowing multiple persons to be associated with a brand. In order to associate each product with *only one* birth year, we would define a *feature* that would compute the average birth year of each individual product, i.e., each product would be associated with the average birth year of its founder(s). This feature could be specified with several methods as described in Section 4.1.2. This feature could also be embedded in a SPARQL query as a *sub-query*; thus a feature can correspond to an inner-query that returns the value of that feature. For instance, in our example we could find at first (i) the average birth year of the founders of each individual product via a sub-query, and then (ii) use the returned results of it in the outer query to calculate the average birth year of all the founders of all the products.

Algorithm 4 Algorithm for computing the components of the translated query for the **General** case

Require: A HIFUN query $q = (gE/rg, mE/rm, opE/ro)$

Ensure: $retVars(gE)$, $opE(mE)$, $triplePatterns(gE)$, $triplePatterns(mE)$, $retVars(gE)$, and $restr(Q_{ans})$

- 1: $triplePatterns(gE) = PairingOverComposition(gE).tp$ ↪triplePatterns for Grouping
 - 2: $retVars(gE) = PairingOverComposition(gE).retVars$ ↪retVars for Grouping
 - 3: **if** $rg \ll \epsilon$ **then** ↪if rg is not empty
 - 4: $triplePatterns(gE).concat(Composition(rg.functions).tp)$ ↪For supporting restrictions by compositions
 - 5: $t \leftarrow last(triplePatterns(gE))$ ↪the last triple pattern of $triplePatterns(gE)$
 - 6: $rgSuffix \leftarrow rg.LastWord$ ↪The last token after the function composition
 - 7: **if** $isURI(rgSuffix)$ **then** ↪Group restriction involving a URI
 - 8: Replace the object of t with $rgSuffix$
 - 9: **else** ↪Group restriction involving a literal
 - 10: Add “FILTER obj(t) $rg.op$ $rgSuffix$ ”
 - 11: $triplePatterns(mE) = Composition(mE).tp$ ↪triplePatterns for Measuring
 - 12: $restr(Q_{ans}) = Composition(mE).retVars$ ↪for the HAVING clause
 - 13: **if** $rm \ll \epsilon$ **then** ↪if rm is not empty
 - 14: $triplePatterns(mE).concat(Composition(rm.functions).tp)$ ↪For supporting restrictions by compositions
 - 15: $t \leftarrow last(triplePatterns(mE))$
 - 16: $rmSuffix \leftarrow rm.LastWord$ ↪The last token after the function composition
 - 17: **if** $isURI(rmSuffix)$ **then** ↪Measuring restriction involving a URI ↪Measuring restriction involving a URI
 - 18: Replace the right variable t with $rmSuffix$
 - 19: **else** ↪Measuring restriction involving a literal
 - 20: Add “FILTER right(t) $rm.op$ $rmSuffix$ ”
 - 21: $opE(mE) = opE(Composition(mE).retVars)$ ↪for the SELECT clause
-

Proposition 1 (Completeness). *The algorithm 4 is complete in the sense that every HIFUN expression can be translated to a SPARQL query.*

Proof. As we have seen in Alg. 4, all parts that a HIFUN expression can have, are considered as input of the algorithm and there is not any restriction in their form.

Proposition 2 (Soundness). *The translation of a HIFUN query to SPARQL according to our method is correct.*

Proof. The semantics of a HIFUN query as defined in [103] are equivalent with the semantics of its translation to SPARQL as defined in [89]. Let q be a HIFUN query and $tr(q)$ be its translation in SPARQL according to our method. We shall prove that the semantics of q are preserved in $tr(q)$ for any KG, i.e. that both have the same answer in any KG.

Let's assume that $q = (productType, price, AVG)$ and $tr(q)$ be its translation in SPARQL.

GROUP BY Translation. In HIFUN, *productType* (i.e. grouping function) has the same semantics as GROUP BY clause in $tr(q)$, iff *productType* is a functional property, i.e., each product has a single type because all components of a HIFUN expression are functional. Thus, since all components in a HIFUN expression are functional, the translation to GROUP BY in SPARQL is valid, iff applied to functional RDF properties.

Aggregate Function Translation: The expression $AVG(price)$ in $tr(q)$ has the same semantics as the *AVG* function applied over the measuring function in HIFUN. Both represent the aggregation of the *price* property. The semantics of $AVG(price)$ in SPARQL align with the semantics for *AVG* aggregation in HIFUN.

Attribute Restriction Translation: Any restriction e in q is translated as a triple pattern $\langle s, p, o \rangle$ in $tr(q)$. For instance, if e is a restriction on a specific HIFUN component i.e. grouping or measuring function, the corresponding triple pattern in $tr(q)$ captures the same semantic condition in SPARQL.

HAVING Restriction Translation: If restriction e in q pertains to the final results, it is translated to the HAVING clause in SPARQL. This ensures that constraints on aggregated values are correctly represented in the translation.



Thus, the translation of a HIFUN query to SPARQL according to our method preserves the semantics of the original HIFUN query across any knowledge graph, confirming the soundness of our translation approach.

Chapter 5

The Proposed Interaction Model for Analytics over RDF

This chapter takes you through the essential aspects of the interaction model for faceted search analytics over RDF data. In Section 5.1, we give a quick overview of the proposed model. Section 5.2 goes into the needed extensions for the formal model, specifically designed to enhance faceted search for advanced analytics over RDF. Moving on to Section 5.3, we explain the interaction model more formally and walk through the associated algorithms. In Section 5.4, we introduce an algorithm designed to implement the state space. Finally, in Section 5.5 we explore the process of expressing and computing intentions embedded in the states.

5.1 The Interaction Model in Brief

GUI Extensions in Brief. The classical FS interface typically consists of two main frames: the left frame for presenting facets and the right frame for displaying objects, illustrated in Figure 5.1 (left). Our proposed model extends the user actions on the left frame by incorporating actions necessary for formulating analytical queries. Notably, it is enriched with two buttons i.e.  and  next to each facet, as shown in Figure 5.1 (right). Additionally, we've introduced an 'Answer Frame' (abbreviated as *AF*) to display the results of analytic queries in tabular or graphical format. To illustrate the concept, we provide below four (4) indicative examples demonstrating how our model facilitates the formulation of both simple and complex analytic queries. We assume that the data of the examples follows the schema of Fig. 1.2.

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. Notably, the condition “US companies” would be specified by expanding the

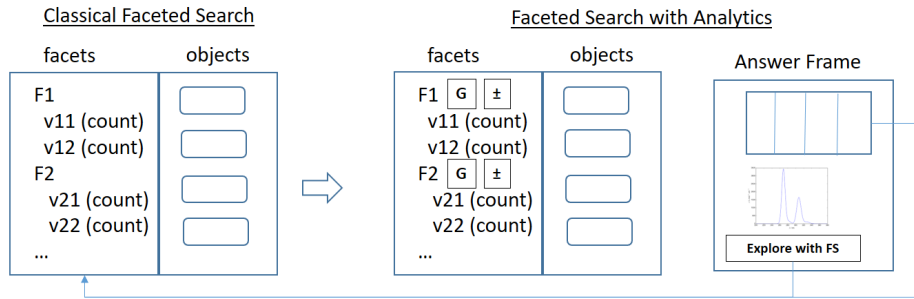


Figure 5.1: The core elements of the GUI for Faceted Search and Analytics

path of the property “manufacturer” till reaching the “origin” property. However, what is missing is the specification of the aggregate function, in this case, “AVG”. To address this, the \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 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’s still missing is the specification of the grouping condition. To address this, 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 adjacent to the “origin” facet. It’s essential to note that, the condition “manufacturers’ country” would be specified by expanding the path of the property “manufacturer” until reaching the “origin” property.

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 a range for the values related to USB ports. To accomplish this, 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 using 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 using the \square and \square buttons laid next to the desired facets, as described in the previous examples. What remains is to restrict the results over the specified threshold. For that, the answer of an analytical query can be loaded as a new dataset onto the extended FS system allowing users to 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 5.2(a). In order to further restrict the answer, we have attached a button, called "Explore with FS" below this table (as shown in Figure 5.1) which lets the users load the results as a new dataset on the FS system as shown in Figure 5.2(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)	(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)

Figure 5.2: Example 3: Exploring the results of an analytic query with faceted search

Expressing the Queries of the Previous Examples in HIFUN. Having described the user interactions, let's show the HIFUN queries formulated during the interaction for the previous examples.

- Example 1: $(\epsilon, price/E, AVG)$, where ϵ 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 $g = (manufacturer \times (\{i \mid i \in D \wedge year \circ releaseDate(i)\}))$, $E = \{i \in D \mid releaseDate(i) = 2021 \wedge origin \circ manufacturer(i) = US \wedge SSD = true \wedge USBPorts(i) = 2\}$ and $F = \{g_i \in (g/E) \mid ans(g_i/E) \geq t\}$

As we can see, expressing queries in a high-level functional query language like HIFUN might pose challenges for an ordinary user, since: (i) expressing queries in a more concise and abstract manner may not be easy, may require documentation and can lead to errors, and (ii) high-level functional query languages may not be as widely adopted or supported as more popular query languages. This can make it harder for users to find help and resources when they encounter issues or need assistance. Overall, while high-level functional query languages can be powerful and flexible tools, they can also require a higher level of expertise and familiarity with functional programming concepts, which can make them more challenging to use effectively.

GUI Extensions. Below, we provide a brief overview of the UI extensions implemented in the classical FS system. These extensions are designed to facilitate the formulation of analytics in HIFUN and to present the results in 2D and 3D graphical formats enhancing their exploration and interpretation.

- **Facets:** On the right side of each facet, there are two buttons: (i) the \mathbb{G} button for grouping the results of the analytical query by this facet and (ii) the \mathbb{F} button that lets users select the function, i.e. avg, min, max, etc., that will be applied to each group of the analytic results,
- **States of \mathbb{G} and \mathbb{F} buttons.** If the user clicks on the \mathbb{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 \mathbb{F} : If the user clicks on the \mathbb{F} 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 (i) tabular or (ii) graphical format.


Tabular results. The analytical results are displayed in a table, since it (i) is familiar to users (ii) offers a clear and concise presentation, and (iii) makes it easy to compare data side by side. However, presenting data in such a format may be dull and unappealing to some readers. It may also result in the data being overlooked or ignored, especially if the table is large or complex, may not be the best choice when presenting complex relationships between data, and may not be practical for displaying very large datasets, as it can become difficult to read and understand.

Graphical results. The analytical results are also displayed both in (a) 2D and (b) 3D plots.

(a) 2D plots: (i) are a widely used and familiar format, so they are easy for people to understand and interpret, (ii) are comprehensive, even for people who are not familiar with the underlying data or analysis, (iii) show relationships between two variables more clearly, and (iv) can be used for many different types of data, including numerical data, categorical data, and time-series data. However, these plots are limited to show relationships between two variables, are not a good choice for showing complex relationships between data, and can be misinterpreted if they are not presented clearly or if the axes are not labeled correctly.

(b) 3D plots: the analytical results are displayed in 3D plots, too, since they: (i) can show relationships between many variables, (ii) can visualize complex relationships between data, (iii) provide more depth than 2D plots, which can make it easier to see patterns and relationships in the data, and (iv) can be more engaging and visually appealing than 2D plots. The 3D visualization of the results are based on a previous work [83] that visualizes the data of the LOD Cloud and the connections them, and its extension¹, that visualizes the progress of COVID-19 virus over time by country. Adopting the metaphor of urban area, each dataset is represented with a multi-store cube. Each segment of the cube corresponds to a feature of the dataset and its volume is proportional to the value of that feature. The proposed system includes interactions and modification of the visualization parameters aiding users to explore the data, extract more details, and create new insights in an intuitive way.

- *Michelanea. Extra Columns.* The answer frame could let users add or remove columns corresponding to the grouping attributes (i.e. display or remove attribute that corresponds to the groups of the results).

Special cases. As stated in [84], there are scenarios where HIFUN is not applicable on a dataset, i.e. if multi-valued attributes or empty values exist. To address such cases, we propose the integration of an additional button, denoted as , next to each facet name that would let users apply *transformation* functions, i.e. the *feature constructor operators*, over it. Such transformations will handle multi-valued attributes or empty values within facets. In addition, such a button would be useful for defining *derived* attributes. For instance, users may leverage it to decompose a date-based attribute into distinct components such as Year, Month, Day, and so forth.

¹<http://62.217.127.128:8080/3dvisualization/>

5.2 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 (Section 5.2.1), and the required extensions of that model for supporting analytics (Section 5.2.2).

5.2.1 Background: The Core Model for FS over RDF

Our approach is based on the general model for Faceted Search over RDF described in [114]. In brief, this model defines the state space of the interaction, where each *state* has an *intention* (query), an *extension* (set of resources), and *transitions* between states facilitated by transition markers (user-clickable elements). 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. Key differentiation of this model, in contrast to classical Faceted Search systems, include (i) its leverage of `rdfs:subClassOf` and `rdfs:subPropertyOf` relations, (ii) its support for the formulation of path expressions (harnessing RDF principles), and the (iii) ability to switch between entity types. These features empower the model to navigate and extract meaningful insights from complex RDF graphs.

5.2.2 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 represent the current state, where $ctx.Ext$ is the set of objects of the current focus (i.e., displayed objects), and $ctx.Int$ is 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, i.e. 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 the parts of the query can be specified through the \mathbb{G} and \mathbb{M} buttons laid next to each facet. In particular,

- clicking on $f.\mathbb{G}$: when the user click on $f.\mathbb{G}$ 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 this value is applied as a restriction on the grouping function.
- clicking on $f.\mathbb{M}$: when the user click on $f.\mathbb{M}$ the intention $ctx.Int$ of the model is changed. Specifically, the measuring function is defined as: $mE' = mE + f$, where f can again de-

note a facet or a property path. If f corresponds to a value, then this value is applied as a restriction on the measuring function.

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. For HAVING queries support, the result set needs to be loaded as a new dataset, enabling users to define analytical queries with unlimited nesting depth. That will also enable users to define analytical queries of unlimited nesting depth.

5.3 The Interaction Model Formally and the Related Algorithms

In this section, we introduce the notations (Section 5.3.1) used to declaratively describe the desired state space of the proposed interaction, (Section 5.3.2). Later, we provide the procedural specification (Section 5.4) and we explain how a result set can be reloaded as a new dataset (Section 5.3.3).

5.3.1 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 C as the set of classes, Pr as the set of properties, \leq_{cl} as the `rdfs:subClassOf` relation between classes, and \leq_{pr} as the `rdfs:subPropertyOf` relation between properties. We also define the instances of a class $c \in C$ as $inst(c) = \{o \mid (o, \text{rdf:type}, c) \in C(K)\}$ and the instances of a property $p \in Pr$ as $inst(p) = \{(o, p, o') \mid (o, p, o') \in C(K)\}$.

For the formal definition of *transitions*, we provide some auxiliary definitions. 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 :

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

5.3.2 Defining the State Space of the Interaction

Below, 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 individuals², or a subset of the dataset, e.g. the result of a keyword query [81], or of a natural language query [80].

Initial Class-based transitions

Below we shall refer to examples assuming the schema of Fig. 1.2, in particular we consider a few instances, specifically the ones illustrated in Fig. 5.3 showing just the most important information.

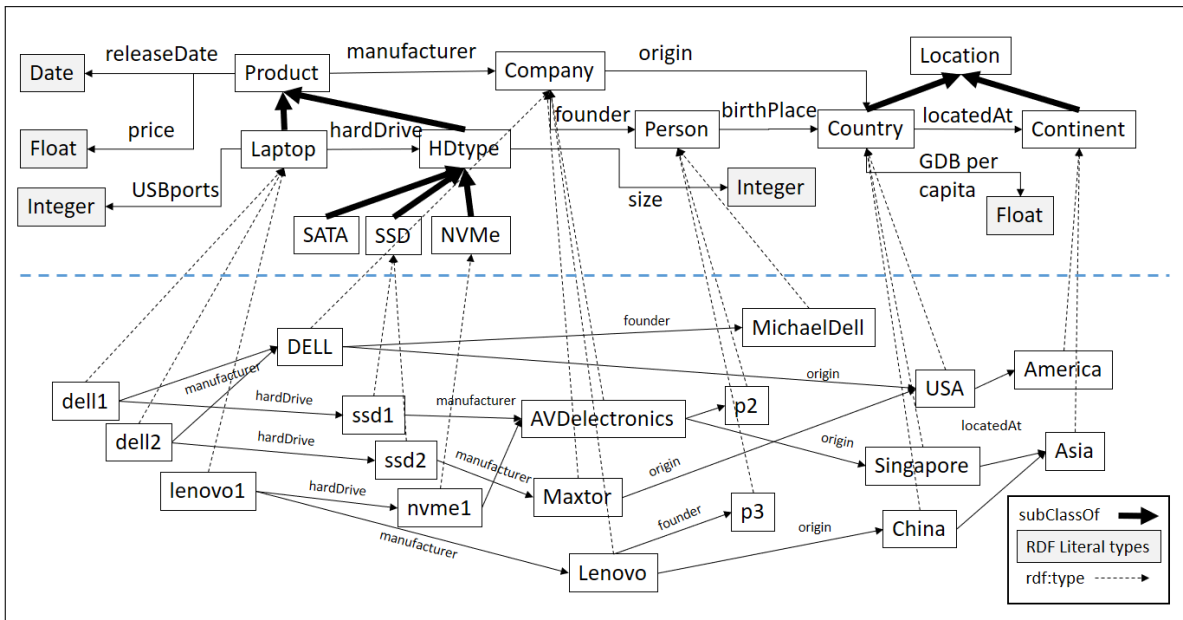


Figure 5.3: Data of our running example

Initially, the top-level or maximal classes $maximal_{\leq cl}(C)$ are presented, as illustrated in Fig. 5.4 (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 to unfold their subclasses, as depicted in Fig. 5.4 (b) as well as their top-level or maximal properties represented by $maximal_{\leq pr}(Pr)$, as shown in Fig. 5.4 (c). Each subclass $c \in subclasses_{\leq cl}(C)$ corresponds to a “class-based transition marker”. Each such transition yields again a state with

²i.e. the results of the SPARQL query ”select ?x where { ?x rdf:type owl:NamedIndividual . }”

(a)	(b)
Company (4) ▣ Location (5) Person (3) ▣ Product (6)	Company (4) ▣ Location (5) Continent (2) Country (3) Person (3) ▣ Product (6) HDType (3) NVMe (1) SSD (2) Laptop (3)
(c)	(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) SSD1 (1) SSD2 (1) NVMe1 (1)	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 5.4: (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,

extension $inst(c)$. If the number of the subclasses is high, then they are hierarchically organized based on the `subClass` relationships among these classes. This hierarchical layout illustrates the structure of the *reflexive and transitive reduction*³ of the *restriction* of \leq_{cl} on the applicable transition markers C , (i.e. on $R^{refl,trans}(\leq_{cl} \upharpoonright C)$). This approach provides a visual representation of the reflexive and transitive reduction, showcasing the hierarchy of transition markers that can be applied.

³The reflexive and transitive reduction of a binary relation R is the smallest relation R' such as both R and R' have the same reflexive and transitive closure.

Property-based transitions

Having expanded the top-level classes, their maximal properties $maximal_{\leq pr}(Pr)$ unfold, too (Fig. 5.4 (c)). 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 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 sub-properties is high, then they are hierarchically organized based on the subproperty relationships among these properties.

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 $1 \leq i \leq k$, 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, \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:

$$M'_i = Restrict(M_i, p_{i+1} : M'_{i+1}) \quad (5.1)$$

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. 5.1 holds also for $i = 0$.

5.3.3 Loading AF as a new Dataset

The results of an analytic query can be loaded as a new derived (RDF) dataset, empowering users to explore and impose further restrictions. 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 subsequent restrictions expressed over this dataset correspond to HAVING clauses over the original data.

(a)	(b)
by manufacturer	by manufacturer (3) ▷ by origin (2)
(2)	DELL (2) US (1)
DELL (2)	Lenovo (1) China (1)
Lenovo (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 manufacturer (2) ▷ by origin
by hardDrive (3)	(2)
SSD1 (1)	SSD1 (1) Maxtor (2) Singapore
SSD2 (1)	(1)
NVMe1 (1)	SSD2 (1) AVDElectronics (1) US (1)
	NVMe1 (1)

Figure 5.5: (a): Property-based transition markers, (b): Property Path-based transitions markers

5.4 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 5.3.2.

5.4.1 Starting Points

The function **Startup** initiates the interaction can start in two ways: (i) from scratch, or (ii) by exploring a set, denoted as *Results*, obtained from an external access method, such as 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 corresponding objects of the right frame.

5.4.2 Computing the Objects in the Right Frame

The computation of the objects in the right frame is detailed in **Part A** of algorithm Alg. 5. The parameter *Filt* can take values such as, "CLASS c", "PVALUE p:v", "PVALUE p:vset", or empty (ϵ). When empty, the current set *E* is defined as all objects in the Knowledge Graph (KG). If nonempty, it restricts the current set of resources *E* according to the notations provided in Section 5.3.1.

Algorithm 5 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: function ComputeNewState(E:set of objects, Filt: Restriction spec, s: current state)
   ↗Part A: Computation of the objects for the right frame
5:   if  $E = \emptyset$  then                            ↗meaning that we have to explore the entire KB
6:      $E \leftarrow Obj$                                 ↗ $E$  is set to the set of all objects
7:   else if  $Filt \neq \epsilon$  then                    ↗if we have to compute a restriction of the current state
8:     switch (Filt):                                ↗Computation of the restricted  $E$  for the right frame
9:       case "CLASS  $c$ ":       $E \leftarrow Restrict(s.Ext, c)$ 
10:      case "PVALUE  $p : v$ ":   $E \leftarrow Restrict(s.Ext, p : v)$ 
11:      case "PVALUE  $p : vset$ ":  $E \leftarrow Restrict(s.Ext, p : vset)$ 
12:     endSwitch
13:   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
14:    $FacetsClasses \leftarrow TM_{cl}(E)$                 ↗The applicable class-based transition markers
15:   for each  $c \in FacetsClasses$  do ↗Creation of the nodes for the transition markers (with names,
   counts, and onClick)
16:     Node  $node \leftarrow new Node()$ ;
17:      $node.name \leftarrow c.name$ ;                    ↗The name that will be displayed
18:      $node.count \leftarrow | Restrict(s.Ext, c) |$       ↗The count that will accompany the name
19:      $node.onClick \leftarrow ComputeNewState(E, "CLASS  $c$ ", s)$  ↗the onClick behavior
20:   Part B.2 see Alg. 6

```

5.4.3 Computing the Facets corresponding to Classes

The part of the algorithm for computing the facets of classes is **Part B.1** of Alg. 5. Being at a state s with extension E , the classes that can be used as class-based transition markers, defined by $TM_{cl}(E)$, are those that the entities in E belong, and they are defined as:

$$TM_{cl}(E) = \{c \in C \mid Restrict(E, c) \neq \emptyset\} \quad (5.2)$$

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., $ComputeNewState(E, "CLASS c ", s)$.



5.4.4 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. 6.

Algorithm 6 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 ← {p ∈ Pr | Joins(s.Ext, p) ≠ ∅}           ↗forward properties
3:   FacetsPropsBack ← {p ∈ Pr-1 | Joins(s.Ext, p) ≠ ∅}       ↗backwards props
4:   for each p ∈ FacetsPropsForw do
5:     Node hnode ← new HeadingNode(p.name)                       ↗Separator and name of p
6:     for each v ∈ 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 ← | Restrict(s.Ext, p : v) |                 ↗The accompanying count
10:      node.onClick ← ComputeNewState(E, "PVALUE p : v", s)
11:      Optional: group all values based on their classes i.e. with respect the following classes:
   {c | Joins(s.Ext, p) ∩ inst(c) ≠ ∅}
12:      hnode.addButton(, onClick=gE+ = p)           ↗For group by
13:      hnode.addButton(, onClick=mE+ = p)           ↗For measuring op
14:      For Entity Type Switch:
15:      hnode.addButton("EntityTypeSwitch",                       ↗For entity type switch
16:        onClick ← ComputeNewState(Joins(s.Ext, p), ε, s))
17:      For Path Expansion:
18:      hnode.addButton("▷",                                       ↗For path expansion
19:        onClick ← ComputeNewState(s.Ext,
20:          "PVALUE p: " + ExpandAndRestrictRecursive(Joins(s.Ext, p)), s)
21:      ...                                                         ↗Analogously for FacetsPropsBack

```

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:

$$P = \{p \in Pr \mid Joins(E, P) \neq \emptyset\} \quad (5.3)$$

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.  and  are created. Finally, an additional expansion button i.e., "▷", is added enabling the user to further expanding the property path (described next in Section 5.4.4).

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*,

“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, (s)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).

Computing the Facets Corresponding to Path Expansion

In Alg. 6, note the line concerning ‘path expansion’ (line 17). On clicking on the element “ \triangleright ” the algorithm for computing the facets that correspond to path expansion is called, detailed in Alg. 7. By clicking on “ \triangleright ” the function “ExpandAndRestrictRecursive(M)” is called again and the process is repeated (as described in Section 5.3.2).

Algorithm 7 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  $\leftarrow \mid Restrict(M, p : v) \mid$                        ✦The accompanying count
9:       node.onClick  $\leftarrow$  return  $Restrict(M, p : v)$                    ✦on click it returns a set
10:      hnode.addButton(">",                                             ✦recursive call for further expansion
11:        onClick  $\leftarrow$  return  $Restrict(M, p, ExpandAndRestrictRecursive(Joins(M, p)))$ 
12:      hnode.addButton( $\mathbb{G}$ , onClick= $gE+ = p$ )                               ✦For group by
13:      hnode.addButton( $\mathbb{H}$ , onClick= $mE+ = p$ )                               ✦For measuring op

```

5.5 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 5.1 (adapted from [114]) 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

Id	Notation	Expression in SPARQL
(i)	$Restrict(E, p : vset)$, where $vset = \{v_1, \dots, v_k\}$	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 $E = \{e_1, \dots, e_k\}$	select Distinct ?v where { ?x p ?v. Filter (?x = e_1 ... ?x = e_k)}
(v)	$TM_c(s.Ext)$ 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 5.1: 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

(i) *extensionally* or (ii) *intentionally*. “Extensional” means that the current state is stored in a temporary class 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 5.2.

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

Table 5.2: 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 fit in the memory, then the intentional approach is preferred. In our implementation, described in the next section, we adopt the intentional approach. Our decision for computing the current state's extension intentionally is driven by the dataset's size and memory constraints. In our implementation, we selectively load data from the triplestore during each state transition, retrieving only the necessary information for that specific state. This strategy optimizes memory usage, allowing the system to handle large datasets effectively while avoiding the need to load the entire dataset into memory at once.

Chapter 6

Implementation

In this section, we delve into our system's architecture, offering a comprehensive view of both its front-end and back-end components (refer to section 6.1). Additionally, we provide a brief overview of the system's core functionality (as detailed in section 6.2).

6.1 Architecture

Regarding the implementation details, we have developed the proposed model into a web-based application.

Server side. The server-side uses the triplestore Virtuoso¹, a high-performance, scalable, and feature-rich RDF triplestore and graph database system. It is often used in applications that need to manage, query, and analyze RDF data. Virtuoso is designed specifically for RDF data, making it an excellent choice for applications deal with semantic data and triples. It (i) supports the SPARQL query language, which is a powerful and standardized way to query RDF data, (ii) can handle large datasets and high query loads, (iii) can be used for data integration purposes, allowing to bring together data from various sources and create a unified RDF dataset (useful if the application needs to combine data from diverse domains), (iv) provides graph database capabilities, i.e. it models and analyzes relationships between entities in RDF data, (v) can serve as a SPARQL endpoint allowing external applications to send SPARQL queries and retrieve RDF data from your triplestore, and (vi) has an active community and commercial support options, providing resources and assistance for developers working with the triplestore. In summary, Virtuoso is a versatile RDF triplestore and graph database system that can be a valuable component in applications dealing with RDF data, semantic web technologies, and complex querying requirements.

Front-end side. The front-end side of the system is implemented in Angular², a popular JavaScript framework that provides a range of benefits for web application develop-

¹<http://docs.openlinksw.com/virtuoso/>

²<https://angular.io/>

ment. Our decision to use Angular was at first, its component-based architecture, which is well-suited for building modular user interfaces, e.g. reusable components for facets, search results, query builders, and more. Since this language is built with TypeScript (which provides strong typing and helps to catch type-related errors at compile time), it would be valuable dealing with structured RDF data ensuring integrity of our data. Also, since Angular offers two-way data binding, it would be easier to reflect changes in the RDF data and query results in real-time on the user interface (which is beneficial for providing a responsive and interactive user experience). Furthermore, Angular incorporates RxJS, which would simplify managing asynchronous data flows and events, making it easier to handle real-time updates, e.g. dealing with RDF data often involves asynchronous operations, such as querying a triplestore. At the same time, this language has a substantial and active community, which means that there is a wealth of resources, documentation, and third-party libraries to help with various aspects of application's development saving time and effort during development. In addition, Angular provides a robust testing framework (Angular Testing Library and Jasmine) that would facilitate unit testing and end-to-end testing of the application, ensuring its reliability and maintainability. Last but not least, the Angular CLI (Command Line Interface) would simplify project setup, development, and deployment tasks. While the current application may be relatively simple, if it grows in complexity over time, Angular's structure and modularity can accommodate that growth more easily than some lighter-weight frameworks.

Figure 6.1 shows the main components of the implementation, and the responsibilities of each component. The User initiates actions by interacting with the knowledge graph explorer, exploring the RDF Knowledge graph and triggering events. The events may lead to (i) requests for exploring further data or (ii) to build analytical queries. In the first case, the user's interactions are sent to the SPARQL query services, the results of which trigger updates in the knowledge graph explorer. In the second case the knowledge graph explorer communicates the events to the analytical query builder informing it of user actions relevant to analytical query construction. The analytical query builder upon receiving user inputs, generates requests for query construction communicating them to the SPARQL query services. Subsequently, the SPARQL query services execute the queries against the RDF Knowledge graph and relay the results to the results display component for rendering.

6.2 System Demonstration

In Fig. 6.2, we present a snapshot of our system, where the loaded data adheres to the schema of the running example. The user has expressed the query: "Average, sum, and maximum prices of laptops with 2 to 4 USB ports, grouped by manufacturer and the manufacturer's origin." To begin with, the results of this query are initially displayed in a tabular

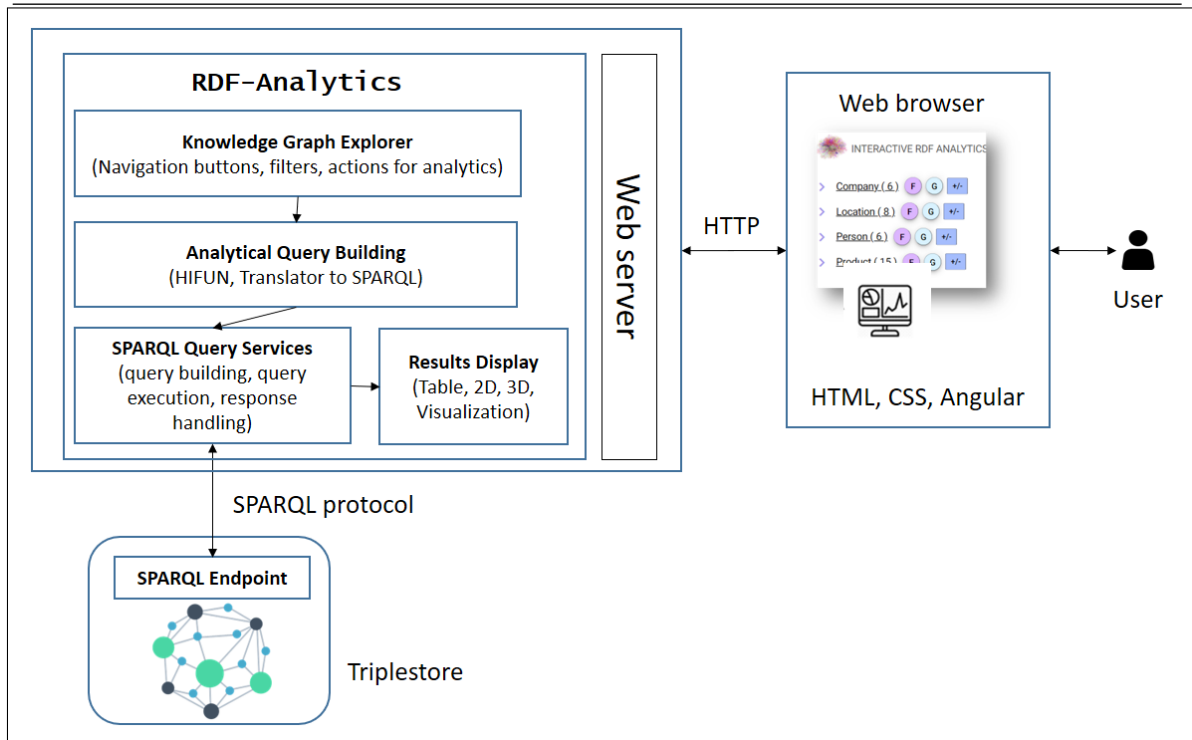


Figure 6.1: The architecture of the system RDF-ANALYTICS

format, as depicted in Fig. 6.3(a). However, our system offers a range of additional options for the user: (i) to further refine the results, users can click on the "Load to Tree" button, resulting in the view illustrated in Fig. 6.3(b), (ii) choose to visualize the results through both two-dimensional and three-dimensional charts for a more comprehensive understanding by selecting the "Visualize" button, as demonstrated in Fig. 6.4, (iii) export the results as a .csv file by simply clicking on the "Export as Excel" button. This multifaceted approach empowers users to interact with and derive more insights from their data.

Common Extensions. The system could benefited from additional 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. Potential extensions include: *search boxes* on each individual facet, methods for *facet/value ranking* (as described in the Section 6.3.2 of the survey [114] and more recent ones like [7, 37]), methods for predicting useful facets [82], as well as with similarity based browsing functions (as in [25]).

6.3 3D Visualization

In addition to presenting analytical results in traditional tabular and two-dimensional formats, we have integrated a 3D visualization system into our framework. This innovative approach, previously developed and described in detail in our earlier work [83]. Within

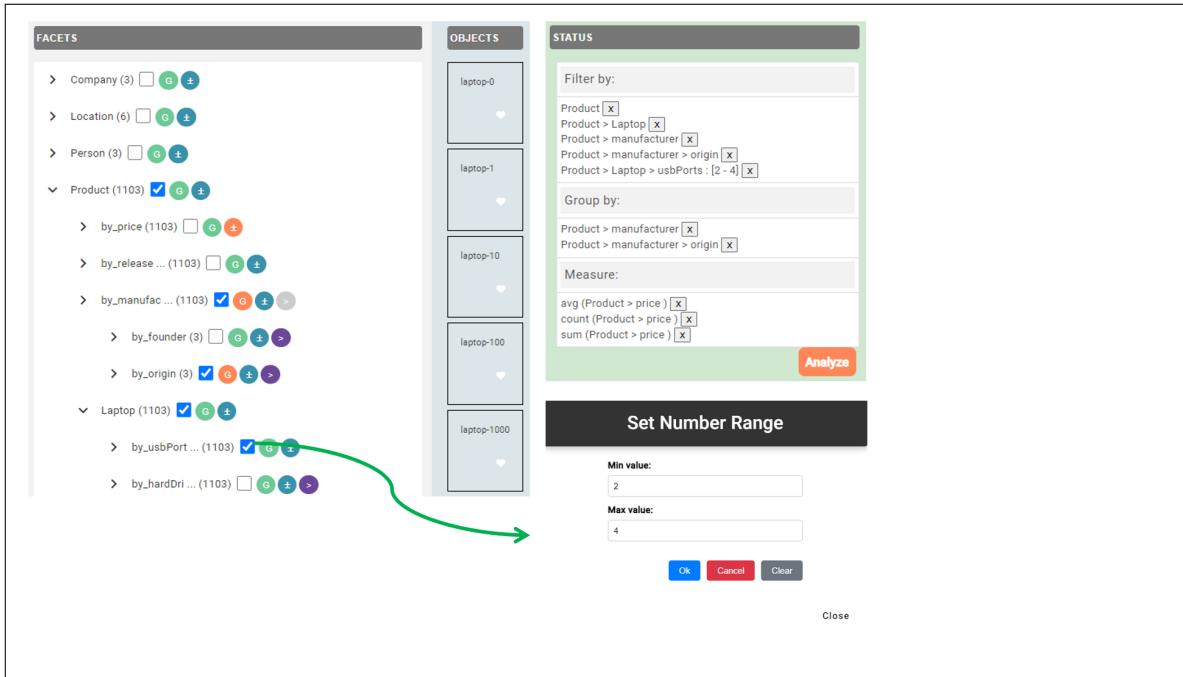


Figure 6.2: GUI of the system RDF-ANALYTICS for formulating the query "Average, sum and max price of laptops that have 2 to 4 USB ports, group by manufacturer and the origin of manufacturer"

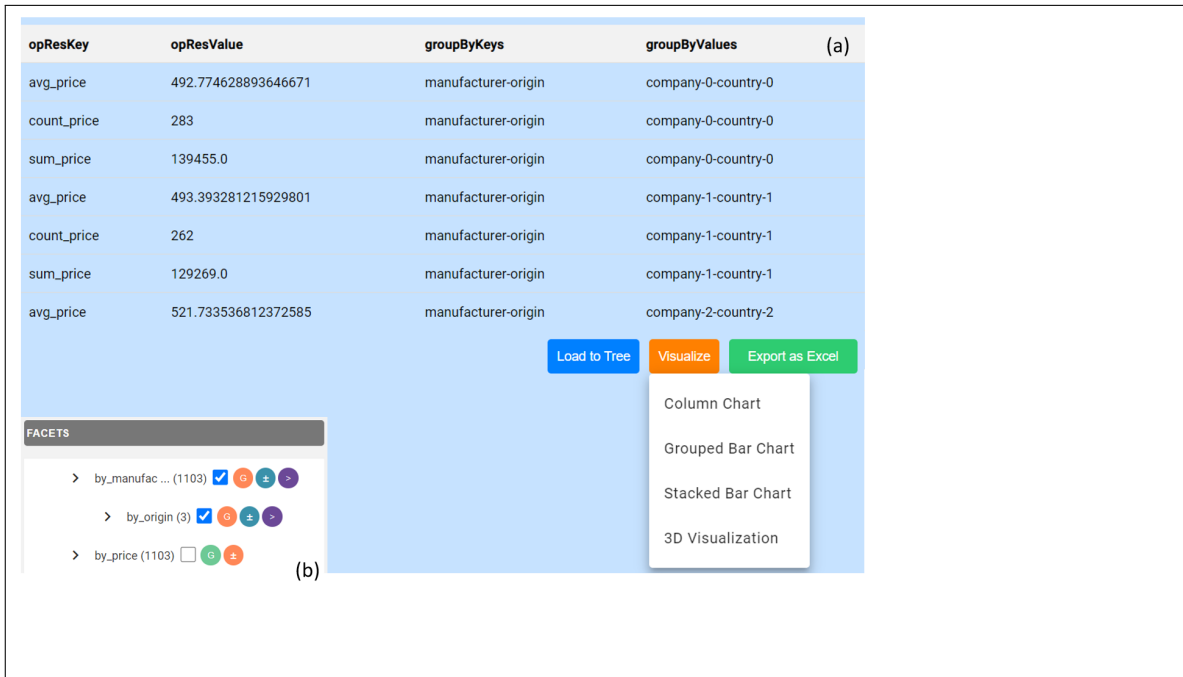


Figure 6.3: (a) Tabular visualization of the results and (b) Loading of the results as a new dataset

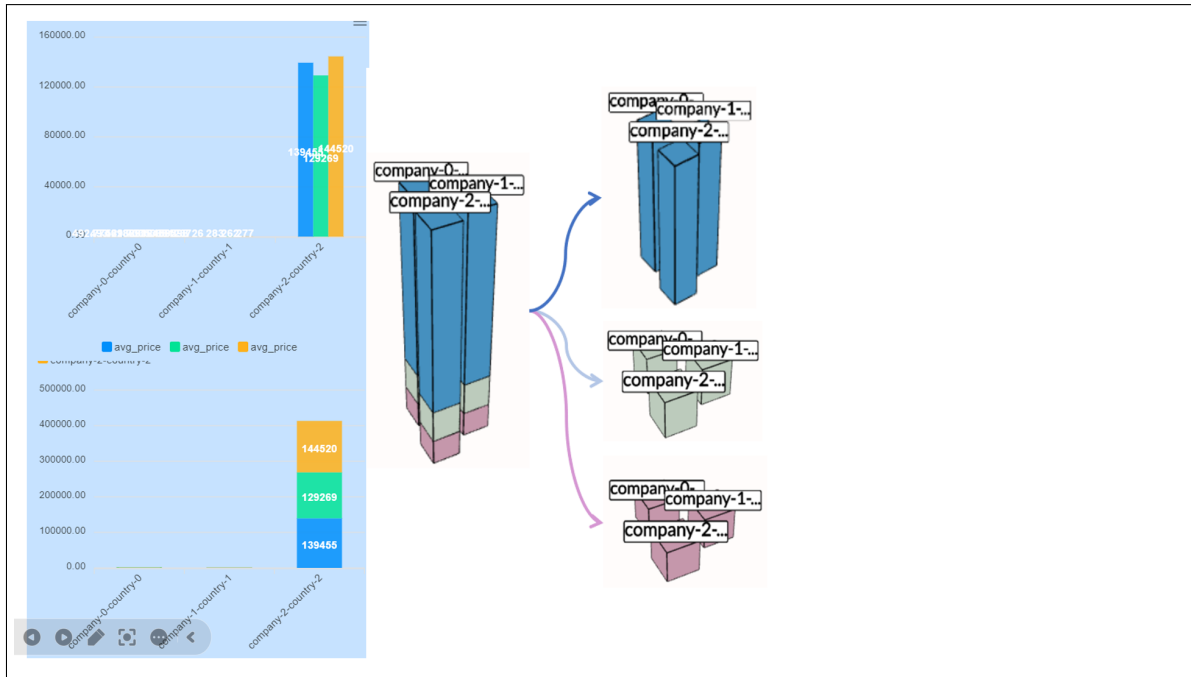


Figure 6.4: 2D and 3D visualization of results

this 3D visualization system, we adopt an urban metaphor, where each numerical result is represented with a 3D building. The volume of the building corresponds to the magnitude of the result, providing an intuitive representation of data values. Furthermore, the color of each building signifies the type of aggregate function applied, such as average, minimum, maximum etc. To enhance clarity, labels associated with these buildings indicate the attributes used for grouping the results. This three-dimensional visualization not only enriches the user experience but also facilitates a deeper understanding of the data, offering an alternative perspective that can reveal patterns and insights that may not be so apparent in conventional 2D representations.

Continuing from our previous example, let's consider the scenario where the results of the query "Average, sum, and maximum prices of laptops with 2 to 4 USB ports, grouped by manufacturer and the manufacturer's origin," are visualized within our 3D system, as illustrated in Fig. 6.5. In this visualization approach, each floor of the 3D building corresponds to a specific aggregate function, such as average (avg), maximum (max), or summation (sum). To enhance clarity, distinct colors are assigned to each floor, providing an immediate visual cue regarding the aggregate function being represented. The system allows also individuals to interactively choose the functions they wish to view within the 3D visualization. By selecting, for instance, only the average, maximum, or sum results, users can easily focus on the specific aspects of interest. This selective rendering capability ensures a tailored and informative visual experience, granting users the flexibility to

explore data insights precisely as they need. Moreover, this feature greatly enhances the interpretability of the results. For instance, users can delve into the average pricing trends (avg) or explore the highest price points (max) for laptops meeting the specified criteria. By visualizing each floor separately, our system empowers users to gain deeper insights into the data, uncovering trends that might otherwise remain hidden in traditional 2D representations.

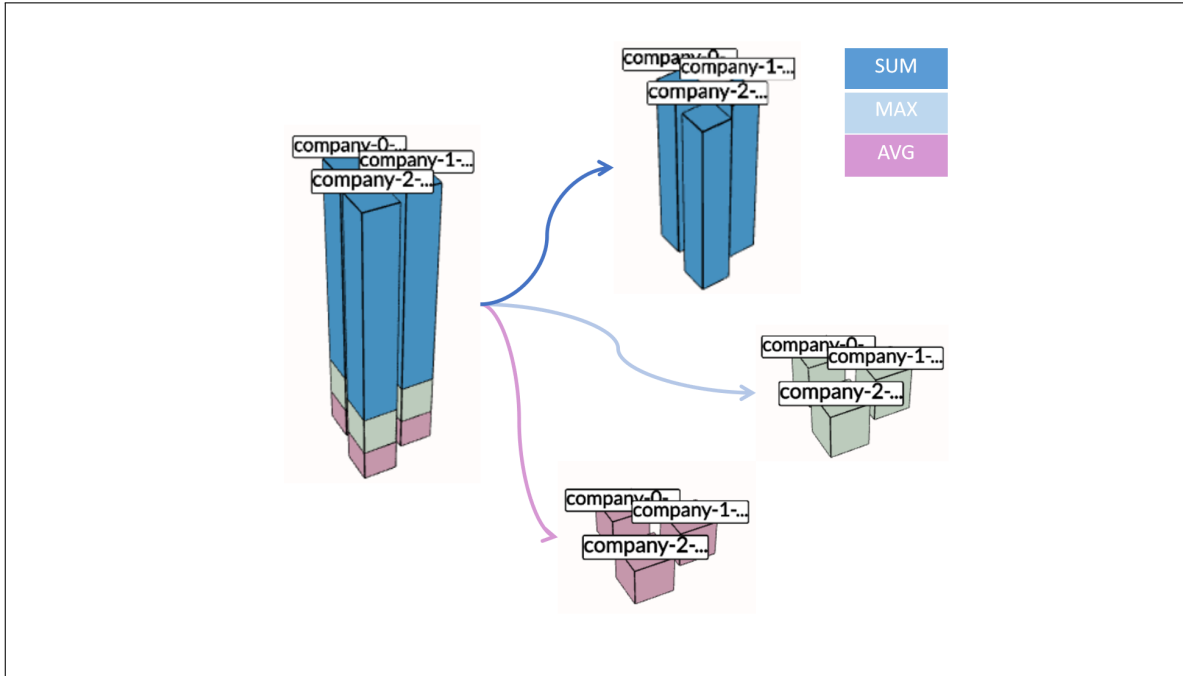


Figure 6.5: 3D visualization of results

6.4 Efficiency

As described in previous section (Section 5.5), there is an extensional and an intentional approach for computing the extension of the current state. The intentional is beneficial for large datasets. Below we discuss the time complexity of the algorithms for computing the required queries following the intentional approach.

Below, we provide a summary of indicative execution times for both simple and more complex queries across datasets of varying sizes. Utilizing the schema of our running example, we generated synthetic datasets ranging from 100 triples to 1 million triples. In the generated RDF datasets, the triples were not unique. To enhance the meaningfulness of aggregate queries and evaluate the efficiency the system effectively, instances were intentionally grouped based on common domains or ranges. These datasets serve as representative samples to evaluate system performance across different data volumes. Subsequently, we

tested the system’s capabilities by executing a diverse set of queries Q1-Q6. These queries encompass a broad spectrum of complexity involving groups, paths, filters, and combinations thereof. It is noteworthy that when executing queries within the dataset containing 1,000,000 triples, we opted not to include instance count information alongside the facets. This decision was made due to the system experiencing performance overload. The significant time investment needed for this counting process contributed to the system’s performance constraints.

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 con

figured with 8.00 GB of RAM and used a 64-bit operating system, specifically Windows 10, version 22H2, on an x64-based processor architecture.

- 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 the last 2 years, 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 the last 2 years, have at least 2 USB ports, and its origin country has gdp per capita [1.5, 2000] group by the origin of manufacturer and the birthplace of the founders (complex query containing paths, filters, groups)

Table 6.1: Efficiency - peak hours

Dataset size (in triples)	Query evaluation (in ms)						Show results (in ms)					
	q1	q2	q3	q4	q5	q6	q1	q2	q3	q4	q5	q6
10^2	149	178	198	316	458	859	28	18	19	16	15	19
10^3	177	188	170	165	362	856	13	57	12	15	10	12
10^4	270	181	319	235	347	576	12	158	20	12	11	12
10^5	147	585	402	240	550	949	16	1005	6	6	7	6
10^6	629	1564	807	930	626	1015	6	1021	6	13	18	5

Based on Table 6.1, several key insights can be drawn regarding query performance. Firstly, it’s evident that the size as the dataset size increases (from 100 to 1,000, 10,000, 100,000, and 1,000,000 triples), the execution times for all queries generally tend to increase. This indicates that dataset size has a significant influence on the time it takes to execute these queries. Larger datasets typically require more processing time. The complexity of the queries also plays a role in execution time. More complex queries, such as Q5

Table 6.2: Efficiency - off-peak hours

Dataset size (in triples)	Query evaluation (in ms)						Show results (in ms)					
	q1	q2	q3	q4	q5	q6	q1	q2	q3	q4	q5	q6
10^2	58	94	212	93	193	758	15	15	12	15	13	14
10^3	75	132	219	94	362	665	13	49	12	13	18	31
10^4	70	167	232	95	362	769	14	175	13	13	13	13
10^5	125	456	290	155	430	781	6	1118	7	6	8	6
10^6	512	1341	682	694	281	567	8	1063	5	8	7	8

and Q6, which involve filtering, grouping, and path traversal, tend to have longer execution times compared to simpler queries like Q1 and Q3. This suggests that query complexity can impact execution time, with more complex queries taking longer to process. The type of query also affects execution time. For example, Q2, which involves grouping by both manufacturer and release date, has varying execution times across dataset sizes, indicating that the combination of groups can influence performance differently compared to simple grouping (Q1). Similarly, the presence of filters (Q3) and path traversal (Q4, Q5, Q6) introduces variations in execution times. At last, the time it takes to display the results is generally much shorter than the query execution time. This suggests that the display of query results is not a significant factor in overall query performance. In summary, dataset size, query complexity, and query type all influence query execution times. Larger datasets generally lead to longer execution times, and more complex queries, involving filtering, grouping, and path traversal, tend to have longer execution times. The specific combination of these factors can result in variations in query performance.

Comparing the set of execution times obtained during off-peak hours of Table 6.2 with the previous results of Table 6.1 it appears that query execution times are shorter compared to the corresponding times during peak hours. The influence of dataset size on query execution times remains consistent with the previous results. As the dataset size increases (from 100 to 1,000, 10,000, 100,000, and 1,000,000 triples), the execution times generally increase, which is expected due to the larger volume of data to process. The impact of query complexity and type on execution time also remains consistent. More complex queries (e.g., Q5 and Q6) continue to have longer execution times compared to simpler queries (e.g., Q1 and Q3). Query complexity, such as filtering, grouping, and path traversal, still affects execution times. In addition, similar to the previous results, the time it takes to display the results remains significantly shorter than the query execution time. This indicates that the display of query results is not a significant factor in overall query performance, and the majority of time is spent on query processing. In conclusion, the new results obtained during off-peak hours generally show shorter query execution times compared to the previous results during peak hours. This suggests that the timing of query execution can have a substantial impact on performance, and conducting

resource-intensive queries during off-peak hours may lead to faster results. However, the overall trends regarding dataset size, query complexity, and query type continue to influence query performance.

While query optimization is not the focus of this work, it's worth noting relevant research in this area. Apart from [8] and [87], notable works include [93] addressing queries that include several chain and star patterns, [126] focusing on optimizing SPARQL queries with aggregate operators, [50] optimizing aggregate queries over federations of SPARQL endpoints by materializing the intermediate results of the queries, and [51] focusing on the selection and materialization of aggregate RDF views for reasons of efficiency, and provides the needed rewriting method for user queries.

Chapter 7

The Expressive Power of the Model

The proposed model aims to cover the most basic information needs in a familiar interactive environment. Providing a model with extreme expressive power would have several disadvantages, including: (i) ambiguity: RDF models used in semantic web applications aim to enable machines to understand the meaning of the data. In these applications, overly expressive models can lead to ambiguity and inconsistency in the data, making it difficult for machines to make sense of it, (ii) interpretation issues: as models become more complex, it becomes increasingly difficult to understand how they are making predictions, which can be a problem in applications where interpretability is important, (iii) high computational cost: overly expressive models in RDF can lead to scalability issues, as the number of triples and the complexity of the relationships between them can grow exponentially. This can make it difficult to process and query the data efficiently.



Below, we describe the expressive power of the proposed model with respect to (a) the kind of HIFUN expressions that can be formulated by the model (in Section 7.1), and (b) the OLAP operations it supports (in Section 7.2).



7.1 Expressible HIFUN queries


Here, we show if the proposed model supports all kind of analytical queries expressed in HIFUN. Recall that, in order to apply HIFUN over a dataset that satisfies its assumptions, we have to specify the context of analysis i.e., the set of attributes we are going to analyze. Then, we are able to formulate queries of the form $(Q = gE/rg, mE/rm, opE/ro)$. As we have already mentioned, the grouping function gE can be a single attribute or a set of complex attributes (pairing or composition) that may have restrictions rg , too. Similarly, the measuring function mE can be a single attribute or a set of complex attributes (pairing or composition) that may have restrictions rm , too. At last, opE can be any valid aggregate function applicable over RDF data. Overall, the model should be able to support complex analytical queries that involve aggregations and filtering operations.



Analysis context. The proposed model lets users define the set of the attributes they are

interested to analyze through the facets it offers, which restrict the information space and specify the focus.

Grouping function. Since a  button lays next to each facet at any level, the user can select to group the results on one or more facets. If the facets correspond to different classes, then gE refers to a *pairing expression*. If a facet corresponds to a property path, then gE refers to a *composition expression*. Additionally, since  button lays next to each facet value, the user can restrict the grouping function on one or more values.

Measuring function. Similarly, since a  button lays next to each facet at any level, the user can select to measure one or more facets. If the facets correspond to different classes, then mE refers to a *pairing expression*. If a facet corresponds to a property path, then mE refers to a *composition expression*. Additionally, since  button lays next to each facet value, the user can restrict the measuring function on one or more values.

Aggregate operation. Since a  button that lays next to each facet offers a menu with the basic aggregation functions, the user can summarize and analyze the results on any such function.

Attribute restrictions. Since the  and  buttons lay next to each distinct value of every facet, the user can restrict the values of the grouping and the measuring function.

Results restrictions. Since the proposed model loads the results of an analytical query as a new dataset on the FS system, the user is able to further explore and restrict them using the filtering capabilities offered by the system.


Nesting. Each time the proposed system loads the current analytical results as a new dataset a new query is created and evaluated. Each such query corresponds to an inner query the results of which are used for the evaluation of an outer query.

7.2 OLAP Operators Supported

Online Analytical Processing systems (OLAP). Online Analytical Processing (OLAP), introduced in the early '90s [26], is used for the analysis of transaction data. It 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 fundamental operations that can be applied over it include: *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).

In our presented model, we enable OLAP-like exploration directly over the graph, without the need of predefining data cubes. The additions we propose for enhancing a Faceted Search (FS) system to support analytic queries effectively cover the requirements for OLAP over graph data. When a user expresses an analytic query with our tool (RDF-ANALYTICS) can achieve the desired granularity and sub-cube is facilitated by just applying restriction and group by actions.

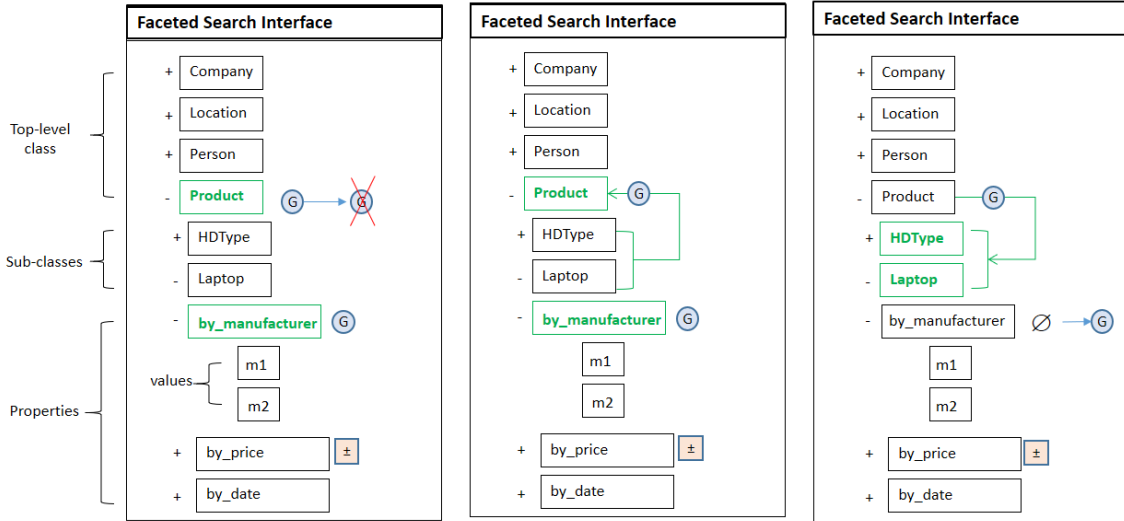
The correspondence between these actions and OLAP operations is illustrated in Figure 7.1. Specifically, traversing up the hierarchy of a facet or adding/removing a GroupBy action corresponds to a roll-up operation. Conversely, traversing down the hierarchy or adding/removing a GroupBy action corresponds to a drill-down operation. Picking one value for a facet corresponds to a slice operation, while picking two or more values from multiple facets corresponds to a dice operation. Finally, moving to a facet directly or indirectly connected to the focus facet corresponds to a pivot operation.

To illustrate this, consider that the user has expressed the query “Average prices for products, grouped by manufacturer,” as shown in Figure 7.2 (left). If the user wishes to drill down and inspect the average prices based on product type, they can press the  button on the class Product. This action corresponds to the query “Average prices of product types grouped by manufacturer,” as depicted in Figure 7.2 (right). The inverse direction would be a roll-up operation.

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

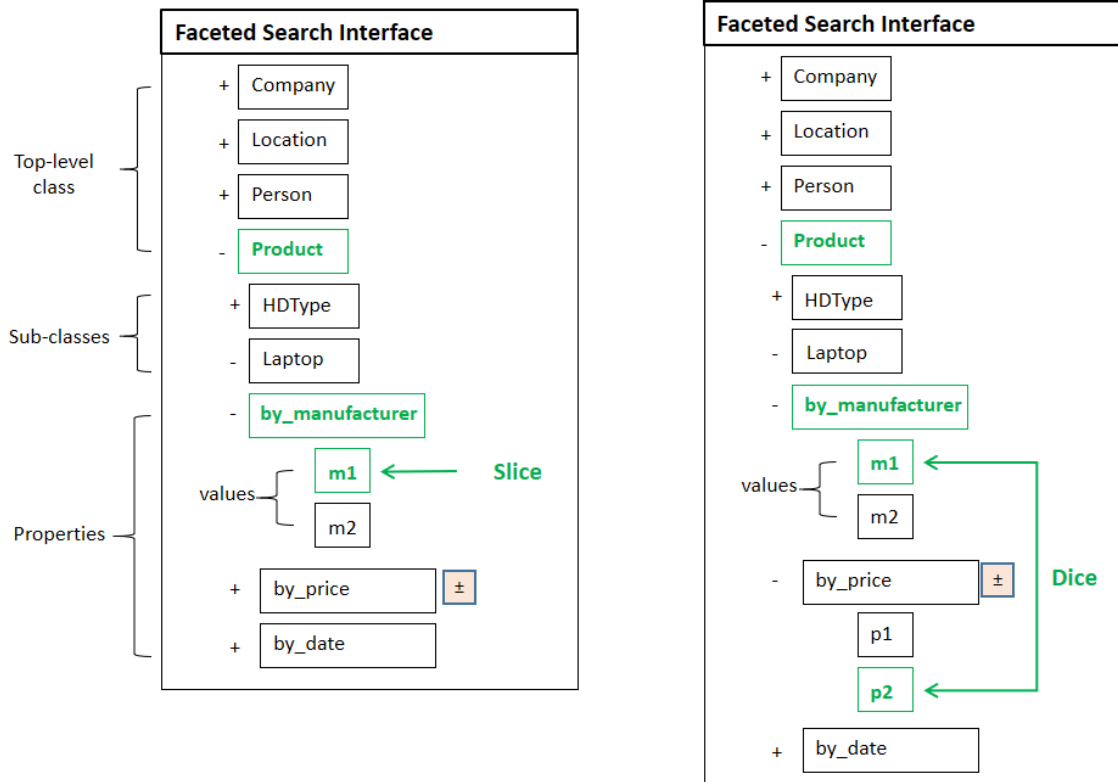


Figure 7.1: Correspondence with OPAP operations

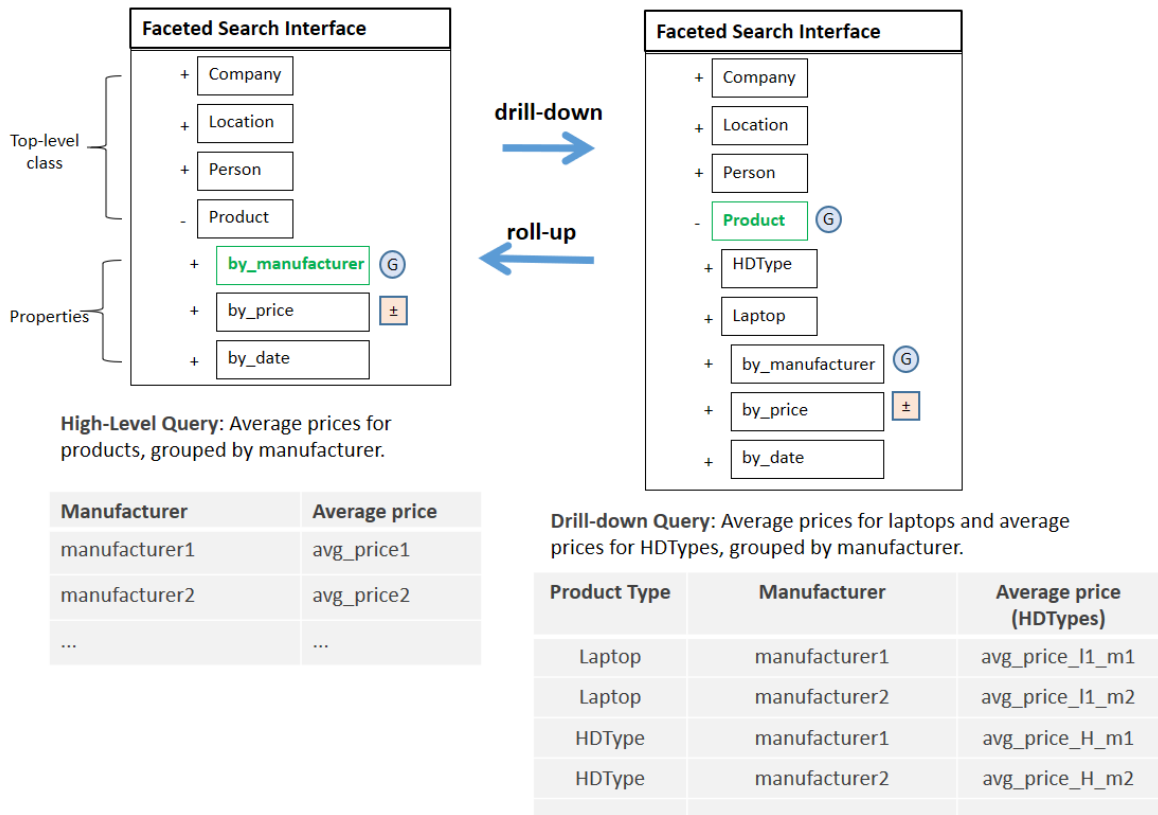


Figure 7.2: An example of roll-up and drill-down

Chapter 8

Evaluation

Section 8.1 discusses the results of an evaluation of our approach with users and Section 8.2 conveys an implementation of the model that proves its feasibility and the completeness of the introduced algorithms.

8.1 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 [36], 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 such as bank employees, accountants and engineers (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 facet¹.

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%)

¹The deployment of the system that was used is accessible at <https://demos.is1.ics.forth.gr/rdf-analytics>.

(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

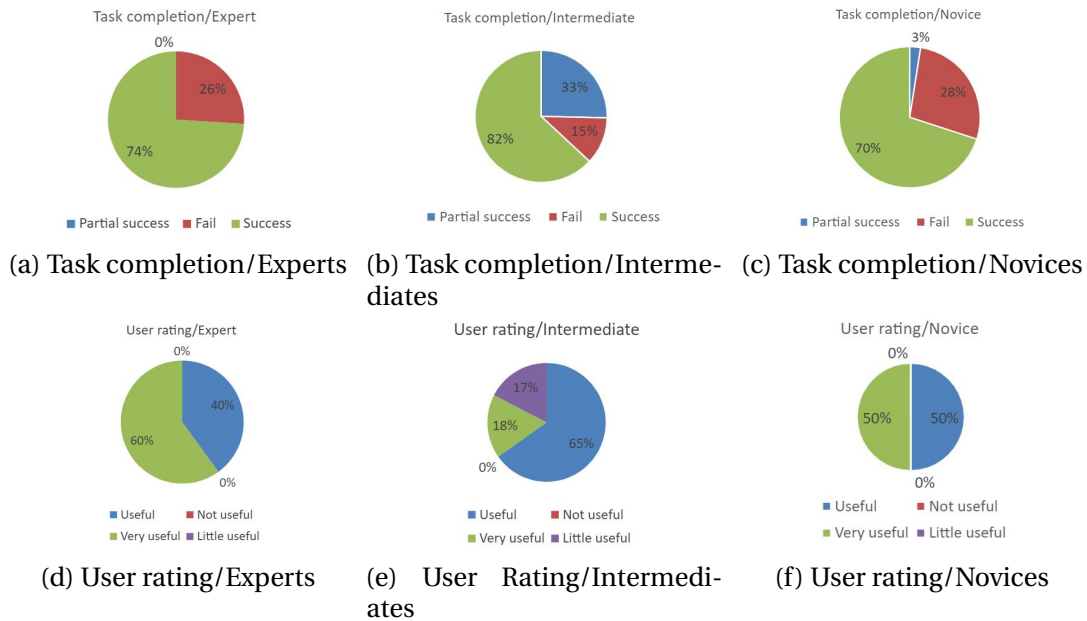


Figure 8.1: Task-based evaluation with users: task completion and user rating

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.

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.

In Fig. 8.2 (a), a notable total success rate of 75.89% is observed, indicating the majority of participants successfully achieved their objectives. However, the presence of a partial success rate of 11.60% suggests that there were instances where participants made notable
















































Figure 8.2: Task-based evaluation with users: total task completion and total user rating

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. 8.2 (b), 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’s 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. 8.1 (a) (i.e., success 75,89%, partial success 11,6%, fail 22,5%) and user rating as shown in Fig. 8.1 (b) (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. \square , \square , \triangleright , 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. , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,

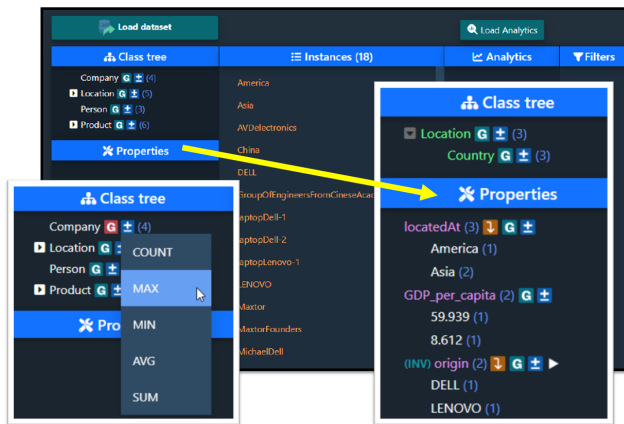


Figure 8.3: An alternative implementation of the proposed model

Chapter 9

Conclusion

The formulation of structured queries over RDF Knowledge Graphs can be challenging, especially when dealing with broad domains containing numerous classes and properties.

To assist both ordinary and expert users, we introduce 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 create an intuitive interaction model and interface, we leverage the familiarity of users with the Faceted Search systems. Started from a general model for Faceted Search over RDF data, we extended it with actions that enable users to specify analytic queries, too. These actions correspond to queries of a high-level language for analytics named HIFUN that we then translate to SPARQL.

Then, we detailed the proposed interaction model by (i) defining formally the state-space of the interaction model and the required algorithms for producing the UI (User Interface) (ii) describing a hybrid (extensional and intentional) query evaluation approach, (iii) presenting an implementation of the model, specifically the system *RDF-ANALYTICS*, that showcased its feasibility, and (iv) discussing the results of a task-based evaluation with users that provided evidence about its acceptance by users in terms of task completion and user rating. In addition, we tested the implementability of the model by third-parties, by showcasing that one undergraduate student succeeded in implementing it from scratch.



Key characteristics of our model are: its applicability to any RDF dataset (not only to RDF datasets that have a star schema), its guidance (by guiding the user to formulate queries whose answer is not empty), the handling of arbitrarily long paths, the provision of count information, the interactive formulation of HAVING clauses, the support for both Faceted Search and analytic queries, and the ability to handle nested analytic queries. As regards visualization, apart from the standard methods to visualize analytic results, we offer a 3D visualization method, too.

Regarding, the results of the task-based evaluation we made with users, they 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 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.

9.1 Future Work and Research

The proposed model has wide applicability, since it can be applied across diverse Knowledge Graphs and complements various methods for accessing graph data. There are several issues for further research and work, a few of them are listed below.

System. Transformations during the Analysis. It is worth investigating the process of introducing *Feature Constructor Operators* (FCO) (as described in Section 4.1.2) to the model and *RDF-ANALYTICS*. These operators will allow users to transform data if needed, at exploration time, to ease analysis. They could be implemented as additional actions, enact-able through a special button (right of the  and  buttons).

System: Ranking of Facets and Terms. In case the KG contains too many maximal classes and maximal properties, as well as if the number of transitions markers is too many, it would be useful to support some *ranking* functions, to avoid cluttering the GUI. One could adopt ideas for such ranking functions from [68, 117].

System: Narrative Visual Analytics, Sequencing Queries, Summaries, and Integration of Large Language Models (LLMs). Users could express their queries, preferences, or exploration goals using natural language. The system would then translate these natural language inputs into structured queries or operations. By integrating sequencing queries, the system could guide users through a step-by-step exploration process, with each query building upon the results of the previous one, contributing to a cumulative understanding. Finally, at certain points or upon user request, the system could provide summaries that distill key insights from the ongoing exploration, acting as checkpoints and offering users a snapshot of their discoveries. The system could also integrate Large Language Models (LLMs). This integration would allow the interpretation of natural language queries as HIFUN queries, enabling users to express their exploration goals in a more intuitive manner. This adaptability would enhance the system's user-friendliness and intuitiveness.

System. Transformation of HIFUN Rewriting Rules to SPARQL Rules. We could optimize the queries expressed in HIFUN through the process of query rewriting. This involves expressing a query in terms of one or more other queries, facilitating the reuse of pre-

evaluated results stored in a cache. As an extension of this effort, the future work could involve the translation of the optimized HIFUN queries into SPARQL fostering a cohesive and efficient querying execution.

System: Further testing and improvement. Looking forward, we are going to evaluate the model's performance on extensive datasets to check its scalability and identify potential optimizations required for handling large-scale data. What we are prioritizing are user-friendly features, intelligibility, and scalability to meet the needs of novice and expert users alike.

System. Extend HIFUN with actions that specify the desired visualization We could investigate enriching HIFUN with actions that specify the desired visualization. In this way, at the stage of results visualization, the system could consider these actions and produce an appropriate visualization

System. Visualization we aim to enrich the model, and the system `RDF-ANALYTICS` with additional visualization features for enhancing the intelligibility of analytical results and offering users a more intuitive understanding of complex data. In particular, we are going to integrate interactive 2D interactive charts, graphs, and diagrams for making insights clearer, allowing users to grasp analytical findings more intuitively and quickly. Furthermore, we are going to add customizable visualization options for enabling users to tailor the presentation of analytical results to their specific needs, thereby enhancing the overall user experience in exploring and interpreting RDF data.

Bibliography

- [1] Ibrahim Abdelaziz, Razen Harbi, Semih Salihoglu, and Panos Kalnis. Combining vertex-centric graph processing with sparql for large-scale rdf data analytics. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3374–3388, 2017.
- [2] Alberto Abelló, Oscar Romero, Torben Bach Pedersen, Rafael Berlanga, Victoria Nebot, Maria Jose Aramburu, and Alkis Simitsis. Using semantic web technologies for exploratory olap: a survey. *IEEE transactions on knowledge and data engineering*, 27(2):571–588, 2014.
- [3] Konrad Abicht, Georges Alkhouri, Natanael Arndt, Roy Meissner, and Michael Martin. Cubviz. js: a lightweight framework for discovering and visualizing rdf data cubes. 2017.
- [4] Bilal Abu-Salih. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076, 2021.
- [5] Akritas Akritidis and Yannis Tzitzikas. Demonstrating interactive SPARQL formulation through positive and negative examples and feedback. In *26th International Conference on Extending Database Technology, EDBT 2023*, 2023.
- [6] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing linked datasets with the void vocabulary. 2011.
- [7] Esraa Ali, Annalina Caputo, Séamus Lawless, and Owen Conlan. Personalizing type-based facet ranking using BERT embeddings, 2021.
- [8] Waqas Ali, Muhammad Saleem, Bin Yao, Aidan Hogan, and Axel-Cyrille Ngonga Ngomo. A survey of RDF stores & SPARQL engines for querying knowledge graphs. *VLDB Journal*, 2021. (accepted for publication).
- [9] Renzo Angles. The property graph database model. In *AMW*, 2018.
- [10] Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter Boncz, George Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan Sequeda, et al. G-core: A core for future graph query languages. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1421–1432, 2018.

- [11] Francesco Antoniazzi and Fabio Viola. Rdf graph visualization tools: A survey. In *2018 23rd Conference of Open Innovations Association (FRUCT)*, pages 25–36. IEEE, 2018.
- [12] Grigoris Antoniou and Frank Van Harmelen. *A Semantic Web Primer*. MIT Press, MA 02142, 2004.
- [13] Marcelo Arenas, Gonzalo I Diaz, and Egor V Kostylev. Reverse engineering SPARQL queries. In *Proceedings of the 25th International Conference on World Wide Web*, pages 239–249, 2016.
- [14] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Šarūnas Marciuška, and Dmitriy Zheleznyakov. Faceted search over RDF-based knowledge graphs. *Journal of Web Semantics*, 37:55–74, 2016.
- [15] Sören Auer, Allard Oelen, Muhammad Haris, Markus Stocker, Jennifer D’Souza, Kheir Eddine Farfar, Lars Vogt, Manuel Prinz, Vitalis Wiens, and Mohamad Yaser Jaradeh. Improving access to scientific literature with knowledge graphs. *Bibliothek Forschung und Praxis*, 44(3):516–529, 2020.
- [16] Elham Akbari Azirani, François Goasdoué, Ioana Manolescu, and Alexandra Roatiş. Efficient OLAP operations for RDF analytics. In *2015 31st IEEE International Conference on Data Engineering Workshops*, pages 71–76. IEEE, 2015.
- [17] Ciro Baron Neto, Kay Müller, Martin Brümmer, Dimitris Kontokostas, and Sebastian Hellmann. Lodvader: An interface to lod visualization, analytics and discovery in real-time. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 163–166, 2016.
- [18] Wouter Beek, Javier D Fernández, and Ruben Verborgh. Lod-a-lot: a single-file enabler for data science. In *Proceedings of the 13th International Conference on Semantic Systems*, pages 181–184, 2017.
- [19] Mohamed Ben Ellefi, Zohra Bellahsene, John G Breslin, Elena Demidova, Stefan Dietze, Julian Szymański, and Konstantin Todorov. Rdf dataset profiling—a survey of features, methods, vocabularies and applications. *Semantic Web*, 9(5):677–705, 2018.
- [20] Boualem Benatallah, Hamid Reza Motahari-Nezhad, et al. Scalable graph-based OLAP analytics over process execution data. *Distributed and Parallel Databases*, 34:379–423, 2016.

- [21] Nikos Bikakis, George Papastefanatos, Melina Skourla, and Timos Sellis. A hierarchical framework for efficient multilevel visual exploration and analysis. *CoRR*, abs/1511.04750, 2015.
- [22] Nikos Bikakis, Melina Skourla, and George Papastefanatos. rdf: Synopsviz—a framework for hierarchical linked data visual exploration and analysis. In *European Semantic Web Conference*, pages 292–297. Springer, 2014.
- [23] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia—a crystallization point for the web of data. *Journal of web semantics*, 7(3):154–165, 2009.
- [24] Šejla Čebirić, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. Summarizing semantic graphs: a survey. *The VLDB journal*, 28(3):295–327, 2019.
- [25] Manos Chatzakis, Michalis Mountantonakis, and Yannis Tzitzikas. RDFsim: similarity-based browsing over DBpedia using embeddings. *Information*, 12(11):440, 2021.
- [26] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. E. F. Codd and Associates, 1993.
- [27] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.
- [28] Gonzalo Diaz, Marcelo Arenas, and Michael Benedikt. SPARQLByE: Querying RDF data by example. *Proceedings of the VLDB Endowment*, 9(13):1533–1536, 2016.
- [29] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. A survey on question answering systems over linked data and documents. *Journal of intelligent information systems*, 55(2):233–259, 2020.
- [30] Dimitar Dimitrov, Erdal Baran, Pavlos Fafalios, Ran Yu, Xiaofei Zhu, Matthäus Zloch, and Stefan Dietze. TweetsCOVID—a knowledge base of semantically annotated tweets about the COVID-19 pandemic. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2991–2998, 2020.
- [31] Syeda Sana e Zainab, Muhammad Saleem, Qaiser Mehmood, Durre Zehra, Stefan Decker, and Ali Hasnain. FedViz: A visual interface for SPARQL queries formulation and execution. In *VOILA@ ISWC*, page 49, 2015.
- [32] Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Hierarchical faceted metadata in site search interfaces. In *CHI’02 Extended Abstracts on Human Factors in Computing Systems*, pages 628–639, 2002.

- [33] Ivan Ermilov, Jens Lehmann, Michael Martin, and Sören Auer. Lodstats: The data web census dataset. In *International Semantic Web Conference*, pages 38–46. Springer, 2016.
- [34] Lorena Etcheverry and Alejandro A Vaisman. QB4OLAP: a new vocabulary for OLAP cubes on the semantic web. In *Proceedings of the Third International Conference on Consuming Linked Data*, volume 905, pages 27–38. CEUR-WS. org, 2012.
- [35] Pavlos Fafalios, Kostas Petrakis, Georgios Samaritakis, Korina Doerr, Athina Kritso-taki, Yannis Tzitzikas, and Martin Doerr. FAST CAT: collaborative data entry and curation for semantic interoperability in digital humanities. *Journal on Computing and Cultural Heritage (JOCCH)*, 14(4):1–20, 2021.
- [36] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35:379–383, 2003.
- [37] Leila Feddoul, Sirko Schindler, and Frank Löffler. Automatic facet generation and selection over knowledge graphs. In *International Conference on Semantic Systems*, pages 310–325. Springer, 2019.
- [38] Alberto Ferrari and Marco Russo. *Introducing Microsoft Power BI*. Microsoft Press, 2016.
- [39] Sébastien Ferré. SPARKLIS: a SPARQL endpoint explorer for expressive question answering. In *ISWC posters & demonstrations track*, 2014.
- [40] Sébastien Ferré. Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web*, 8(3):405–418, 2017.
- [41] Sébastien Ferré. Analytical queries on vanilla RDF graphs with a guided query builder approach. In *International Conference on Flexible Query Answering Systems*, pages 41–53. Springer, 2021.
- [42] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1433–1445, 2018.
- [43] Ningchao Ge, Peng Peng, Zheng Qin, and Mingdao Li. Fedaggs: Optimizing aggregate queries evaluation in federated rdf systems. In *International Conference on Web Information Systems Engineering*, pages 527–535. Springer, 2021.

- [44] Olaf Görlitz and Steffen Staab. Splendid: Sparql endpoint federation exploiting void descriptions. *COLD*, 782, 2011.
- [45] Armin Haller, Javier D Fernández, Maulik R Kamdar, and Axel Polleres. What are links in linked open data? a characterization and evaluation of links between knowledge graphs on the web. *Journal of Data and Information Quality (JDIQ)*, 12(2):1–34, 2020.
- [46] SM Shamimul Hasan, Donna Rivera, Xiao-Cheng Wu, Eric B Durbin, J Blair Christian, and Georgia Tourassi. Knowledge graph-enabled cancer data analytics. *IEEE journal of biomedical and health informatics*, 24(7):1952–1967, 2020.
- [47] Ali Hasnain, Qaiser Mehmood, Syeda Sana e Zainab, and Aidan Hogan. Sportal: profiling the content of public sparql endpoints. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 12(3):134–163, 2016.
- [48] Patrick Hoefler, Michael Granitzer, Vedran Sabol, and Stefanie Lindstaedt. Linked data query wizard: A tabular interface for the semantic web. In *Extended Semantic Web Conference*, pages 173–177. Springer, 2013.
- [49] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4):1–37, 2021.
- [50] Dilshod Ibragimov, Katja Hose, Torben Bach Pedersen, and Esteban Zimányi. 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*, pages 269–285. Springer, 2015.
- [51] Dilshod Ibragimov, Katja Hose, Torben Bach Pedersen, and Esteban Zimányi. 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*, pages 341–359. Springer, 2016.
- [52] Antoine Isaac and Bernhard Haslhofer. Europeana linked open data–data. european.eu. *Semantic Web*, 4(3):291–297, 2013.
- [53] Kim Ahlstrøm Jakobsen, Alex B Andersen, Katja Hose, and Torben Bach Pedersen. Optimizing rdf data cubes for efficient processing of analytical queries. In *COLD*, 2015.

- [54] Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246, 2019.
- [55] Maulik R Kamdar and Mark A Musen. Phlegra: Graph analytics in pharmacology over the web of life sciences linked open data. In *Proceedings of the 26th International Conference on World Wide Web*, pages 321–329, 2017.
- [56] Evgeny Kharlamov, Luca Giacomelli, Evgeny Sherkhonov, Bernardo Cuenca Grau, Egor V Kostylev, and Ian Horrocks. Semfacet: Making hard faceted search easier. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2475–2478, 2017.
- [57] Jakub Klímek, Jiří Helmich, and Martin Nečaský. Payola: Collaborative linked data analysis and visualization framework. In *Extended Semantic Web Conference*, pages 147–151. Springer, 2013.
- [58] Yuta Kobayashi, Hiroyuki Shindo, and Yuji Matsumoto. Scientific article search system based on discourse facet representation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9859–9860, 07 2019.
- [59] Mikko Koho, Esko Ikkala, Petri Leskinen, Minna Tamper, Jouni Tuominen, and Eero Hyvönen. Warsampo knowledge graph: Finland in the second world war as linked open data. *Semantic Web Interoperability, Usability, Applicability*, 2020. In press.
- [60] Vangelis Kritsotakis, Yannis Roussakis, Theodore Patkos, and Maria Theodoridou. Assistive query building for semantic data. In *SEMANTICS Posters&Demos*, 2018.
- [61] Petri Leskinen, Goki Miyakita, Mikko Koho, and Eero Hyvönen. Combining faceted search with data-analytic visualizations on top of a sparql endpoint. In *CEUR Workshop Proceedings*, 2018.
- [62] Hao Li, Chee-Yong Chan, and David Maier. Query from examples: An iterative, data-driven approach to query construction. *Proceedings of the VLDB Endowment*, 8(13):2158–2169, 2015.
- [63] Alexander Loth. *Visual analytics with Tableau*. John Wiley & Sons, 2019.
- [64] Pierre Maillot, Olivier Corby, Catherine Faron, Fabien Gandon, and Franck Michel. KartoGraphI: Drawing a Map of Linked Data. In *ESWC 2022 - 19th European Semantic Web Conferences*, Hersonissos, Greece, May 2022. Springer.

- [65] Eetu Mäkelä. Aether–generating and viewing extended void statistical descriptions of rdf datasets. In *European Semantic Web Conference*, pages 429–433. Springer, 2014.
- [66] Mohamed Nadjib Mami, Damien Graux, Harsh Thakkar, Simon Scerri, Sören Auer, and Jens Lehmann. The query translation landscape: a survey. *arXiv preprint arXiv:1910.03118*, 2019.
- [67] Paolo Manghi, Alessia Bardi, Claudio Atzori, Miriam Baglioni, Natalia Manola, Jochen Schirrwagen, Pedro Principe, Michele Artini, Amelie Becker, Michele De Bonis, et al. The openaire research graph data model. *Zenodo*, 2019.
- [68] Kostas Manioudakis and Yannis Tzitzikas. Faceted search with object ranking and answer size constraints. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–33, 2020.
- [69] Michael Martin, Konrad Abicht, Claus Stadler, Axel-Cyrille Ngonga Ngomo, Tommaso Soru, and Sören Auer. Cubeviz: Exploration and visualization of statistical linked data. In *Proceedings of the 24th International Conference on World Wide Web*, pages 219–222, 2015.
- [70] Franck Michel, Fabien Gandon, Valentin Ah-Kane, Anna Bobasheva, Elena Cabrio, Olivier Corby, Raphaël Gazzotti, Alain Giboin, Santiago Marro, Tobias Mayer, et al. Covid-on-the-Web: Knowledge graph and services to advance COVID-19 research. In *International Semantic Web Conference*, pages 294–310. Springer, 2020.
- [71] Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. Loupe-an online tool for inspecting datasets in the linked data cloud. In *ISWC (Posters & Demos)*, 2015.
- [72] José Moreno-Vega and Aidan Hogan. GraFa: Scalable faceted browsing for RDF graphs. In *International Semantic Web Conference*, pages 301–317. Springer, 2018.
- [73] Michalis Mountantonakis, Carlo Allocca, Pavlos Fafalios, Nikos Minadakis, Yannis Marketakis, Christina Lantzaki, and Yannis Tzitzikas. Extending void for expressing connectivity metrics of a semantic warehouse. In *PROFILES@ ESWC*, 2014.
- [74] Michalis Mountantonakis and Yannis Tzitzikas. How linked data can aid machine learning-based tasks. In *International Conference on Theory and Practice of Digital Libraries*, pages 155–168. Springer, 2017.
- [75] Michalis Mountantonakis and Yannis Tzitzikas. LODsyndesis: global scale knowledge services. *Heritage*, 1(2):23, 2018.

- [76] Michalis Mountantonakis and Yannis Tzitzikas. Large-scale semantic integration of linked data: A survey. *ACM Computing Surveys (CSUR)*, 52(5):103, 2019.
- [77] Michalis Mountantonakis and Yannis Tzitzikas. Lodchain: Strengthen the connectivity of your rdf dataset to the rest lod cloud. In *International Semantic Web Conference*, pages 537–555. Springer, 2022.
- [78] Michalis Mountantonakis and Yannis Tzitzikas. 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*, 2023.
- [79] Lekha Nair, Sujala Shetty, and Siddhanth Shetty. Interactive visual analytics on big data: Tableau vs d3. js. *Journal of e-Learning and Knowledge Society*, 12(4), 2016.
- [80] Christos Nikas, Pavlos Fafalios, and Yannis Tzitzikas. Open domain question answering over knowledge graphs using keyword search, answer type prediction, SPARQL and pre-trained neural models. In *International Semantic Web Conference*, pages 235–251. Springer, 2021.
- [81] Christos Nikas, Giorgos Kadilierakis, Pavlos Fafalios, and Yannis Tzitzikas. Keyword search over RDF: Is a single perspective enough? *Big Data and Cognitive Computing*, 4(3):22, 2020.
- [82] Xi Niu, Xiangyu Fan, and Tao Zhang. Understanding faceted search from data science and human factor perspectives. *ACM Transactions on Information Systems (TOIS)*, 37(2):1–27, 2019.
- [83] Maria-Evangelia Papadaki, Panagiotis Papadakos, Michalis Mountantonakis, and Yannis Tzitzikas. An interactive 3D visualization for the LOD cloud. In *EDBT/ICDT Workshops*, pages 100–103, 2018.
- [84] Maria-Evangelia Papadaki, Nicolas Spyrtatos, and Yannis Tzitzikas. Towards interactive analytics over RDF graphs. *Algorithms*, 14(2):34, 2021.
- [85] Maria-Evangelia Papadaki and Yannis Tzitzikas. Rdf-analytics: Interactive analytics over rdf knowledge graphs. 2023.
- [86] Maria-Evangelia Papadaki and Yannis Tzitzikas. Unifying faceted search and analytics over rdf knowledge graphs. Manuscript under review, 2023.
- [87] Maria-Evangelia Papadaki, Yannis Tzitzikas, and Michalis Mountantonakis. A brief survey of methods for analytics over RDF knowledge graphs. *Analytics*, 2(1):55–74, 2023.

- [88] Maria-Evangelia Papadaki, Yannis Tzitzikas, and Nicolas Spyrtatos. Analytics over RDF graphs. In *International Workshop on Information Search, Integration, and Personalization*, 2019.
- [89] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics of sparql, 2008.
- [90] Ruben Prieto-Diaz. Implementing faceted classification for software reuse. *Communications of the ACM*, 34(5):88–97, 1991.
- [91] F. Gandon R. Gazzotti, F. Michel. Cord-19 named entities knowledge graph (cord19-nekg), 2020.
- [92] C Rajeswari, Dyuti Basu, and Namita Maurya. Comparative study of big data analytics tools: R and tableau. In *IOP Conference Series: Materials Science and Engineering*, volume 263, page 042052. IOP Publishing, 2017.
- [93] Padmashree Ravindra, Vikas V Deshpande, and Kemafor Anyanwu. Towards scalable RDF graph analytics on mapreduce. In *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*, pages 1–6, 2010.
- [94] Chavva Subba Reddy, Ravi Sankar Sangam, and B Srinivasa Rao. A survey on business intelligence tools for marketing, financial, and transportation services. In *Smart intelligent computing and applications*, pages 495–504. Springer, 2019.
- [95] Tony Russell-Rose and Tyler Tate. *Designing the search experience: The information architecture of discovery*. Elsevier, 2013.
- [96] Giovanni Sacco. Dynamic taxonomies: A model for large information bases. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):468–479, 2000.
- [97] Giovanni Maria Sacco and Yannis Tzitzikas. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*. Springer, Verlag, 2009.
- [98] Ahmad Sakor, Samaneh Jozashoori, Emetis Niazmand, Ariam Rivas, Kostantinos Bougiatiotis, Fotis Aisopos, Enrique Iglesias, Philipp D Rohde, Trupti Padiya, Anastasia Krithara, et al. Knowledge4covid-19: A semantic-based approach for constructing a covid-19 related knowledge graph from various sources and analysing treatments’ toxicities. *Journal of Web Semantics*, page 100760, 2022.
- [99] Percy E Rivera Salast, Michael Martin, Fernando Maia Da Mota, Sören Auer, Karin K Breitman, and Marco A Casanova. Olap2datacube: An ontowiki plug-in for statistical data publishing. In *2012 Second International Workshop on Developing Tools as Plug-Ins (TOPI)*, pages 79–83. IEEE, 2012.

- [100] Evgeny Sherkhonov, Bernardo Cuenca Grau, Evgeny Kharlamov, and Egor V Kostylev. Semantic faceted search with aggregation and recursion. In *International Semantic Web Conference*, pages 594–610. Springer, 2017.
- [101] Arnaud Soulet and Fabian M Suchanek. Anytime large-scale analytics of linked open data. In *International Semantic Web Conference*, pages 576–592. Springer, 2019.
- [102] Nicolas Spyratos. A functional model for data analysis. In *International Conference on Flexible Query Answering Systems*, 2006.
- [103] Nicolas Spyratos and Tsuyoshi Sugibuchi. HIFUN-a high level functional query language for big data analytics. *Journal of Intelligent Information Systems*, 51:529–555, 2018.
- [104] Nicolas Spyratos and Tsuyoshi Sugibuchi. Data exploration in the HIFUN language. In *Flexible Query Answering Systems: 13th International Conference, FQAS 2019, Amantea, Italy, July 2–5, 2019, Proceedings 13*, pages 176–187. Springer, 2019.
- [105] Radhika Sridhar, Padmashree Ravindra, and Kemafor Anyanwu. Rapid: Enabling scalable ad-hoc analytics on the semantic web. In *International Semantic Web Conference*, pages 715–730. Springer, 2009.
- [106] Minna Tamper, Petri Leskinen, Eero Hyvönen, Risto Valjus, and Kirsi Keravuori. Analyzing biography collections historiographically as linked data: Case national biography of finland. *Semantic Web*, (Preprint):1–35, 2022.
- [107] Bogaard Tessel. Metadata categorization for identifying search patterns in a digital library. 75(2):270–286, Jan 2019.
- [108] Yannis Theoharis, Yannis Tzitzikas, Dimitris Kotzinos, and Vassilis Christophides. On graph features of semantic web schemas. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):692–702, 2008.
- [109] Ilaria Tiddi and Stefan Schlobach. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence*, 302:103627, 2022.
- [110] Paul Town and Fadi Thabtah. Data analytics tools: A user perspective. *Journal of Information & Knowledge Management*, 18(01):1950002, 2019.
- [111] Gerwald Tschinkel, Eduardo E Veas, Belgin Mutlu, and Vedran Sabol. Using semantics for interactive visual analysis of linked open data. In *ISWC (Posters & Demos)*, pages 133–136, 2014.
- [112] Daniel Tunkelang. *Faceted search*, volume 5. Morgan & Claypool Publishers, USA, 2009.

- [113] Yannis Tzitzikas. FS2KG: From file systems to knowledge graphs (demo). In *ISWC 2022*, 2022.
- [114] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. Faceted exploration of RDF/S datasets: a survey. *Journal of Intelligent Information Systems*, 48(2):329–364, 2017.
- [115] Yannis Tzitzikas, Yannis Marketakis, Nikos Minadakis, Michalis Mountantonakis, Leonardo Candela, Francesco Mangiacrapa, et al. 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*, pages 20–34. Springer, 2019.
- [116] Yannis Tzitzikas, Maria-Evangelia Papadaki, and Manos Chatzakis. A spiral-like method to place in the space (and interact with) too many values. *Journal of Intelligent Information Systems*, pages 1–25, 2021.
- [117] Yannis Tzitzikas and Panagiotis Papadakos. Interactive exploration of multi-dimensional and hierarchical information spaces with real-time preference elicitation. *Fundamenta Informaticae*, 122(4):357–399, 2013.
- [118] Dylan Van Assche, Thomas Delva, Gerald Haesendonck, Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. Declarative rdf graph generation from heterogeneous (semi-) structured data: A systematic literature review. *Journal of Web Semantics*, page 100753, 2022.
- [119] Oskar van Rest, Sungpack Hong, Jinha Kim, Xuming Meng, and Hassan Chafi. Pqql: a property graph query language. In *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems*, pages 1–6, 2016.
- [120] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledge-base. *Communications of the ACM*, 57(10):78–85, 2014.
- [121] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Douglas Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, et al. Covid-19 open research dataset (cord-19), 2020.
- [122] Gerhard Weikum. Knowledge graphs 2021: a data odyssey. *Proceedings of the VLDB Endowment*, 14(12):3233–3238, 2021.
- [123] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. DrugBank 5.0: a major

- update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.
- [124] Airton Zancanaro, LD Pizzol, RDM Speroni, José Leomar Todesco, and FAO Gauthier. Publishing multidimensional statistical linked data. In *Proceedings of the Fifth International Conference on Information, Process, and Knowledge Management*, pages 290–304, 2013.
- [125] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph cube: on warehousing and OLAP multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 853–864, 2011.
- [126] Lei Zou, M Tamer Özsu, Lei Chen, Xuchuan Shen, Ruizhe Huang, and Dongyan Zhao. gstore: a graph-based sparql query engine. *The VLDB journal*, 23(4):565–590, 2014.

