Federated Query Processing over Heterogeneous Data Sources in a Semantic Data Lake

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Kemele M. Endris
aus
Durbete, Äthiopien

Bonn, 15.11.2019

Dieser Forschungsbericht wurde als Dissertation von der Mathematisch-Naturwissenschaftlichen Fakultät der Universität Bonn angenommen und ist auf dem Hochschulschriftenserver der ULB Bonn `https://nbn-resolving.org/urn:nbn:de:hbz:5-57937` elektronisch publiziert.

1. Gutachter:	Prof. Dr. Sören Auer
2. Gutachter:	Prof. Dr. Jens Lehmann
Tag der Promotion:	03.03.2020
Erscheinungsjahr:	2020

Abstract

Data provides the basis for emerging scientific and interdisciplinary data-centric applications with the potential of improving the quality of life for citizens. Big Data plays an important role in promoting both manufacturing and scientific development through industrial digitization and emerging interdisciplinary research. Open data initiatives have encouraged the publication of Big Data by exploiting the decentralized nature of the Web, allowing for the availability of heterogeneous data generated and maintained by autonomous data providers. Consequently, the growing volume of data consumed by different applications raise the need for effective data integration approaches able to process a large volume of data that is represented in different format, schema and model, which may also include sensitive data, e.g., financial transactions, medical procedures, or personal data. Data Lakes are composed of heterogeneous data sources in their original format, that reduce the overhead of materialized data integration. Query processing over Data Lakes require the semantic description of data collected from heterogeneous data sources. A Data Lake with such semantic annotations is referred to as a Semantic Data Lake. Transforming Big Data into actionable knowledge demands novel and scalable techniques for enabling not only Big Data ingestion and curation to the Semantic Data Lake, but also for efficient large-scale semantic data integration, exploration, and discovery. Federated query processing techniques utilize source descriptions to find relevant data sources and find efficient execution plan that minimize the total execution time and maximize the completeness of answers. Existing federated query processing engines employ a coarse-grained description model where the semantics encoded in data sources are ignored. Such descriptions may lead to the erroneous selection of data sources for a query and unnecessary retrieval of data, affecting thus the performance of query processing engine. In this thesis, we address the problem of federated query processing against heterogeneous data sources in a Semantic Data Lake. First, we tackle the challenge of knowledge representation and propose a novel source description model, RDF Molecule Templates, that describe knowledge available in a Semantic Data Lake. RDF Molecule Templates (RDF-MTs) describes data sources in terms of an abstract description of entities belonging to the same semantic concept. Then, we propose a technique for data source selection and query decomposition, the MULDER approach, and query planning and optimization techniques, Ontario, that exploit the characteristics of heterogeneous data sources described using RDF-MTs and provide a uniform access to heterogeneous data sources. We then address the challenge of enforcing privacy and access control requirements imposed by data providers. We introduce a privacy-aware federated query technique, BOUNCER, able to enforce privacy and access control regulations during query processing over data sources in a Semantic Data Lake. In particular, BOUNCER exploits RDF-MTs based source descriptions in order to express privacy and access control policies as well as their automatic enforcement during source selection, query decomposition, and planning. Furthermore, BOUNCER implements query decomposition and optimization techniques able to identify query plans over data sources that not only contain the relevant entities to answer a query, but also are regulated by policies that allow for accessing these relevant entities. Finally, we tackle the problem of interest based update propagation and co-evolution of data sources. We present a novel approach for interest-based RDF update propagation that consistently maintains a full or partial replication of large datasets and deal with co-evolution.

Contents

CHAPTER 1

Introduction

In the last decade, the amount of data available has exponentially grown [1, 2] and it is expected a faster growth rate in the next years as a result of the progress in existing devices for data generation and ingestion. Furthermore, FAIR principles and open data initiatives have encouraged the publication of Big Data by exploiting the decentralized nature of the Web and allowing, thus, for the availability of heterogeneous data generated and maintained by autonomous data providers. Heterogeneity can be on different levels, i.e., syntactic, semantic, access interface, and data model. The autonomous nature of data providers generate the heterogeneity of data sources for several reasons. For instance, different data providers model the same real-world concepts differently, even though, data is represented using single data model such as relational model [3]. In addition, some applications require certain technical environments, data models, or features such as performance for different types of queries. The number of data sources available on the Web presents a wealth of data and, at the same time, a challenge for processing them in a meaningful way.

A data integration system provides a uniform access to heterogeneous, autonomous, and distributed data sources. There are two approaches for data integration from disparate data sources, materialized (data warehousing) and virtual data integration. Traditional data warehouses ingest data from heterogeneous data sources to a centralized database after transforming them, using Extract-Transform-Load (ETL) processes following *schema-on-write* paradigm, into a common structure. Transformation of disparate data into single data model eventually might degrade the performance of the query processing, suffer from freshness problem, and may lead eventually to information silos. In virtual data integration approaches data stay at the sources in their original format; data is accessed at query time allowing on-the-fly transformation of heterogeneous data. Given a query formulated in a global access interface, a virtual data integration system extracts and combines results from different sources to answer the query. Data sources on the Web are autonomous in a way that they evolve independently and may not be available at anytime or have limited query capabilities. Thus, data integration system should be able to adapt to the conditions of the data sources. Federated query processing engines are a type of virtual data integration systems that deal with such autonomous data sources.

Available data sources may have different data structuredness, querying capabilities, and access interfaces and restrictions. Sensitive data that is restricted to only authorized users, for instance in biomedical, government, and financial data, must not be revealed. Such regulations should be respected during data access from these data sources as well as during aggregation of data with other sources. For these reasons, relevant sources should be selected at query time based on a description about their content and capabilities. Description of such heterogeneous data sources is crucial for different applications, such as data source discovery, analysis, and integration. In the context of big data, the role of *data source*

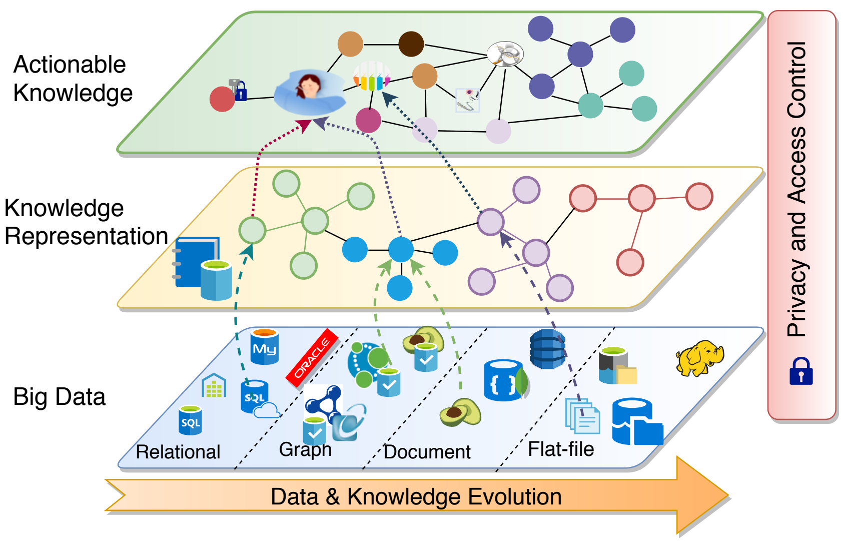


Figure 1.1: **From Big Data to Actionable Knowledge:** At lowest layer, heterogeneous big data distributed in data sources. Raw data can be in different data models, such as Relational: where data is stored in a structured way via tables, Graph: where data is stored in terms of nodes and their relationships, Document (JSON-like): where data is stored in terms of collections, and Flat files: stored in local file system or data lake stores like Hadoop, Azure or S3. At the middle layer, the raw data at lowest layer is semantically lifted via mapping to domain knowledge via ontologies and vocabularies. Finally, at the top layer a virtually integrated actionable knowledge can be extracted. In each layers, privacy and access control policies of the data sources should be respected. Furthermore, data and knowledge evolves through time. The knowledge extraction should also consider these changes as well

descriptions is important for making heterogeneous data sources interoperable as well as for efficient and effective integration, and management of data sources.

To provide scalable and flexible knowledge discovery, analysis, and reporting, *Data Lakes* have been proposed [4, 5]. Data Lakes compose heterogeneous data sources in their original format, following *schema-on-read* paradigm. They reduce the cost of identifying, storing, cleansing, and integrating data significantly and promote flexibility in data analysis. Data Lakes introduce complexity during query execution, as data is stored in raw format and different interoperability issues can arise between data sources. Big Data systems that integrate different data sources need to handle variety, volume, and volatility of data efficiently and effectively. The publication of a large number of data on the Web fosters the development of query processing infrastructures to access this data in a federated fashion. Federated query processing techniques guarantee the freshness of data accessed directly from the data providers. For instance, SPARQL endpoints provide Web interfaces to access RDF data following SPARQL protocol. Federated query processing engines are able to merge data extracted from such distributed data sources. The role of semantic annotation of data sources is important in doing so. *Semantic Data Lakes* are proposed to include a semantic layer that provide semantic annotations [6]. The goal of this thesis is to provide federated query processing technique over heterogeneous data sources in a Semantic Data Lake while enforcing access control and privacy policies imposed by data providers, and managing the data and knowledge evolution when a replica or slice of data is created.

1.1 Motivation

Big data plays an important role in promoting both manufacturing and scientific development through industrial digitization and emerging interdisciplinary research. However, the availability of enormous amount of data generated in scientific and industrial domains demands the development of computational methods for exploration and analysis, as well as transformation of big data to actionable knowledge. Although a rich variety of tools and big data collections are available, many challenges need to be addressed in order to discover insights from which decisions can be taken. For instance, different interoperability conflicts can exist among data collections, data may be incomplete, and entities may be dispersed across different datasets. Furthermore, data providers impose privacy and access control regulations while processing data points. For example, a cancer research center can be allowed to perform certain operations on specific data points about patients without revealing the identity of the patient. These issues hinder knowledge exploration and discovery, being thus required data integration in order to unveil meaningful outcomes.

Consider, for example, a set of data sources about the condition of a lung cancer patient, as well as typical data integration problem caused as a result of data complexity issues, e.g., variety, volume, veracity. Electronic health records (EHRs) preserve the knowledge about the conditions of a patient that need to be considered for effective diagnoses and treatment prescriptions. However, such pieces of knowledge is usually stored in different formats, e.g., relational tables that store patient demographics, or flat files stores gnomic analysis, liquid biopsies, or clinical notes. This dispersed data needs to be integrated in a meaningful way to get the complete information about conditions of a patient. Furthermore, physicians depend on their experience or available sources of knowledge to predict potential adverse outcomes, e.g., drug interactions, side-effects or resistance. Diverse repositories and databases make available crucial knowledge for the complete description of a patient condition and the potential outcome. Nevertheless, sources are autonomous and utilized diverse formats that range from unstructured scientific publications to structured data about cancer related mutations. In order to detect facts that can impact on the effectiveness of a particular treatment, e.g., Docetaxel, a physician will have to search through these diverse data sources and identify the potential adverse events and interactions. Data complexity issues like data volume and diversity impede an efficient integration of the knowledge required to predict the outcomes of a treatment. Transforming big data into actionable knowledge demands novel and scalable tools for enabling not only big data ingestion and curation, but also for efficient large-scale semantic data integration, exploration, and discovery.

Semantic data integration approaches allows at generating common representation of concepts and their relations using domain knowledge formalisms in the form of ontologies and reasoning capabilities offered by the Semantic Web technologies. Figure 1.1 illustrates the need for semantic data integration from heterogeneous data sources. Information about a patient, for example, might be spread across different data sources, such as hospital patient database, drug interactions (DrugBank[1]) and side effects (SIDER[2]) on the Web, and EHRs as flat files. The first layer, Big Data, comprises of data sources that represent data in its original format and contains implicit facts. Different data models are used to represent raw data, such as relations, graph, document, and flat files. Different data management systems are used to store data, which supports different access methods, query language, and processing capabilities. The second layer, Knowledge Representation, represents knowledge available in the data sources by explicit semantic labeling of concepts and their relationships. For instance, information dispersed in different data sources that corresponds to patient information can be represented as a formal concept `onto:Patient`. Rule-based mappings can be used to create mappings between the data elements to ontology concepts, lifting raw data to RDF knowledge representation. Given a formal knowledge representation, different data integration techniques can be applied to produce actionable knowledge, i.e., third layer. Such techniques exploit the semantics encoded in representation of data during processing.

[1] `https://www.drugbank.ca/`
[2] `http://sideeffects.embl.de/`

Different applications, such as question answering and data analytic, can make use of such knowledge which in turn can be applied for decision making. During data retrieval and processing in each layer, privacy and access control regulations imposed by data providers needs to be respected. For example, when aggregating data about the patient from hospital database with drug side effects on the Web, sensitive information about the patient should not be transferred outside the hospital network or revealed to the user. Finally, data and knowledge evolves through time. Access to fresh data is crucial in most applications. Knowledge about the concepts and relation between them could be added or removed, changing the knowledge about the entities, conversely changing the implicit knowledge represented in them. On the other hand, facts about instances of concepts change. Those changes need to be dealt with whenever they occur. In order to provide a unified access to heterogeneous, autonomous, and distributed data sources, challenges in each layer need to be tackled. In the following section, we discuss the challenges that motivate this thesis.

1.2 Problem Statement and Challenges

The decentralized and autonomous nature of data publishers allows for data being produced and represented in different forms. Federated query processing techniques integrate data from autonomous, distributed, and heterogeneous data sources in a uniform way by minimizing query execution time while maximizing answer completeness. Given a federated query in a formal language, such as SPARQL, against a federation of heterogeneous data sources, the problem of federated query processing is to: (i) efficiently and effectively execute queries over data sources, (ii) enforces privacy and access control policies during query execution, and (iii) collects and merge data from heterogeneous data sources in the federation. In scenarios where accessing data directly from the provider is not feasible, the datasets can be replicated, either the entire dataset or subset (slice) of it. In such cases, a replicated (target) data source need to (co)evolve as the original data source evolves. Given a replica of a data source and an update interest description, the problem of co-evolution is to propagate only important, i.e., interesting, updates from source to target, and vice-versa guaranteeing the freshness of data and knowledge. Figure 1.2 illustrates three-layer and two orthogonal dimensions to transform big data to actionable knowledge and the challenges that motivate this thesis.

Challenge 1: Describe Knowledge available in Heterogeneous Data Sources

Federated query processing techniques rely on descriptions of data sources. Data source descriptions guide the source selection, query decomposition, and planning techniques of the federated query processing engines to provide an efficient and effective execution plan. Such descriptions facilitate federated query engines to discover relevant data sources for processing and to interpret them accordingly. Moreover, these descriptions can be used to analyze data source features and conform design patterns set during knowledge graph generation. Several data source description vocabularies are adopted by the Semantic Web community (e.g., DCAT [3], DCMI [4], and VoID [5]). Though, these descriptions have been used for federated query processing over RDF data sources, they are not scalable and expressive enough to be used in the context of heterogeneous data sources. On the other hand, the database community adopts view-based data source descriptions by defining global schema (mediated schema). Data source description can be categorized into content and access level description. Content descriptions represent

[3] https://www.w3.org/TR/vocab-dcat/
[4] www.dublincore.org/specifications/dublin-core/dces/
[5] https://www.w3.org/TR/void/

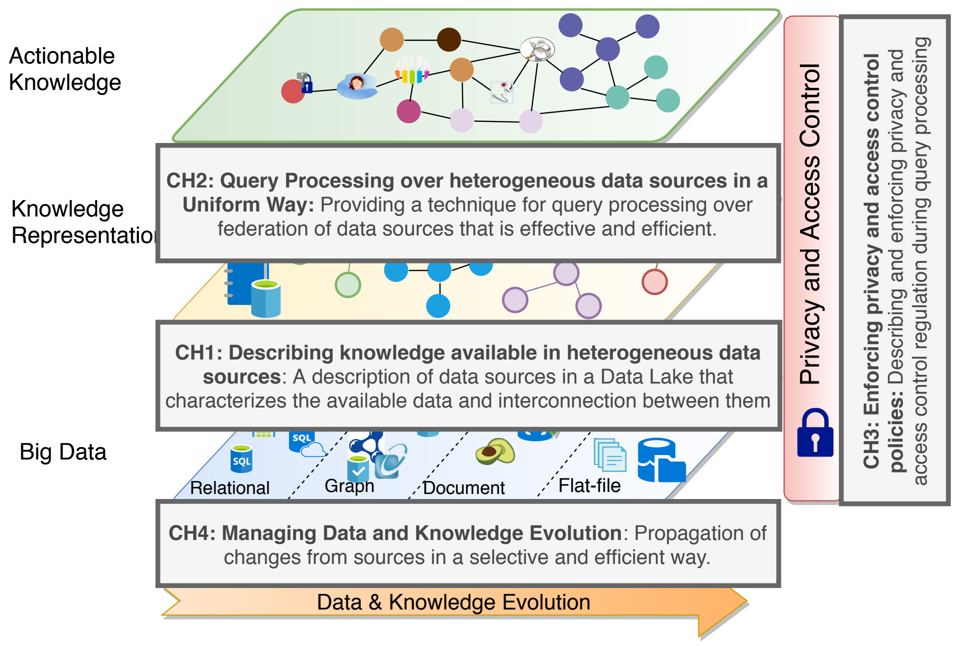

Actionable Knowledge
Knowledge Representation
Big Data
CH2: Query Processing over heterogeneous data sources in a Uniform Way: Providing a technique for query processing over federation of data sources that is effective and efficient.
CH1: Describing knowledge available in heterogeneous data sources: A description of data sources in a Data Lake that characterizes the available data and interconnection between them
Relational
Graph
Document
Flat-file
CH4: Managing Data and Knowledge Evolution: Propagation of changes from sources in a selective and efficient way.
Data & Knowledge Evolution
Privacy and Access Control
CH3: Enforcing privacy and access control policies: Describing and enforcing privacy and access control regulation during query processing

Figure 1.2: **Challenges.** To transform big data to actionable knowledge, we need to tackle four main challenges: **CH1** – Describing knowledge available in heterogeneous data sources, **CH2** – Query processing over heterogeneous data sources in a uniform way, **CH3** – Enforcing privacy and access control policies, and **CH4** – Managing Data and Knowledge Evolution

the semantic description of entities available in the data sources, while access descriptions represent the privacy and access policies, access interfaces, and querying mechanisms. The challenges include: extracting the semantics encoded in the data sources, capturing the relationship between concepts (semantic) in different sources, and enabling representation of other information, such as access restrictions/patterns, and update expressions. Data source descriptions that are able to express the semantics encoded in heterogeneous data sources is crucial for efficient and effective processing of queries over the federation and to handle the evolving nature of data and knowledge.

Challenge 2: Query Processing over Heterogeneous Data Sources in a Uniform Way

Federated query processing techniques need to combine data from variety of data sources in a uniform way by minimizing query execution time while at the same time maximizing answer completeness. Approaches that provide a flexible solution to the problem of query processing over a federation of heterogeneous data sources are crucial. Since the number of potentially relevant data sources for a query can be very large, one of the major challenges of the query engines is the selection of minimal number of sources that can provide the data required to answer a query [3]. An efficient big and heterogeneous, data management, and query processing techniques are crucial. Existing approaches are not able to exploit knowledge about the main features of the integrated data sources, and produce query plans customized for sources selected for collecting the data from sources in the federation. Selecting relevant data sources for a specific query, creating an efficient query execution plan considering data source types (capabilities), and combining partial results obtained from these sources are the main challenges in query processing over heterogeneous data sources.

Challenge 3: Enforcing Privacy and Access Control Policies during Query Processing

Effective data centric applications demand data management techniques able to process a large volume of data which may include sensitive data, e.g., financial transactions, medical procedures, or personal data. Managing sensitive data requires the enforcement of privacy and access control regulations, particularly, during the execution of queries against datasets that include sensitive and non-sensitive data. The challenge in enforcing privacy and data access policies are two fold: first the engine has to ensure only relevant data sources are selected to evaluate the given query, and second it has to guarantee sensitive data is not revealed while merging data from different data sources that may contain sensitive data. Most of the existing work focuses on the specification of access control ontologies and their enforcement on centralized or distributed data stores. Albeit expressive, these approaches are not able to consider privacy and access control regulations during the whole pipeline of a federated query processing, i.e., source selection, query decomposition, planning, and execution. As a consequence, efficient query plans cannot be devised in a way that privacy-aware policies are enforced. Merging sensitive data requires the enforcement of privacy and access regulations imposed by the provider at source level as well as at mediator level, i.e., at the query engine. Data source providers might allow certain operations to be performed on certain data points either at the premise of the provider or at mediator level without revealing sensitive content. Such requirements require a more expressive description model for privacy and access policy specification. Moreover, enforcing privacy and access regulations add an overhead during execution of queries. Federated query processing approaches need to generate valid plans that respect the privacy and access policies and minimize execution time. Selecting relevant data sources that can contribute to answer the query without violating privacy and access regulations imposed by data providers, generating valid plans by selecting appropriate algorithms that respect privacy and access regulations set by the data providers, and minimizing execution time and maximize answer completeness with respect to privacy and access policy restrictions are the main challenges in enforcing privacy and access policies during federated query processing.

Challenge 4: Managing Evolution of Data and Knowledge

A number of data sources are available on the Web where users can send requests for specific data using access interfaces provided by the data providers. Many data providers serve large amount of requests from diverse applications, and many data products and services rely on full or partial data available in those sources. Due to limited resources to serve a large amount of requests from different applications, data providers enforce a restriction on the number of results, query patterns, and number of requests within a period of time. Hence, many products and services rely on full or partial data replications to ensure reliable federated query processing and overcome those restrictions which do not met their application requirements. Given the evolving nature of the original and authoritative datasets, to ensure consistency and freshness, replicas need to be replaced frequently. Such frequent replacement for every update might become impractical, especially for data sources that have frequent data changes such as sensor data, social networks, etc. Moreover, all updates of the original data source might not be of interest for some applications or simply do not have enough resources to replicate the whole dataset by propagating all updates. Thus, only interesting changes should be retained while propagating updates to the replicas. On the other hand, the replicas might evolve independently of the original source, which might lead to inconsistent data between the source and target (i.e., replica) datasets if same data point is edited. Challenges when propagating updates include the expression of important (*interesting*) updates, propagation of updates that the application is interested in, and resolution of conflicts that might arise if the target dataset is allowed to evolve.

1.3 Research Questions

The following research questions are defined in the scope of this thesis based on the challenges identified in the previous section.

> RQ1: How can we describe the semantics encoded in heterogeneous data sources?

To answer this research question, we investigate the state-of-the-art data source description and profiling techniques used by federated query processing engines. We adopt the definition of RDF molecules and their associated semantic types and properties for source description. RDF molecules are defined as a set of triples that share the same subject. We provide a set of metrics from graph theory to describe certain features of data sources and demonstrate the applicability by describing state-of-the-art benchmarks for RDF data federation. Finally, we investigate the effect of different source description techniques for query processing and evaluate their effect on the performance of a federated query engine. In the context of this research question, we assume data source are either inherently represented in RDF data model or the rule-based mapping from non-RDF data model to ontology is available.

> RQ2: How can features represented in data source descriptions be employed to guide the federated query processing over heterogeneous data sources?

In order to address this research question, first we analyze state-of-the-art semantic data integration techniques, in particular, federated query processing techniques that employ SPARQL as a global querying language. Based on our analysis, we define query decomposition and source selection techniques that exploit a novel source description model and evaluate our technique over different benchmarks. We compare the proposed query decomposition and source selection technique against federation of data sources with state-of-the-art federated query engines. Then, we investigate the behavior of the optimizer in the context of heterogeneous data sources and propose query planning and optimization techniques in the presence of heterogeneity. We evaluate the effectiveness and efficiency of query processing technique using the source descriptions that represent the semantics encoded in the data sources.

> RQ3: How can privacy and access control requirements be encoded in data source descriptions and be used for enforcement during query processing over federation of data sources?

To answer this research question, we study privacy and access control restriction description methods in the literature. We investigate different access control requirements that a federated query processing engine has to respect in a distributed environment where sensitive and non-sensitive open data shared between different authoritative entities. We investigate the effect of the access control and privacy policy enforcement on the performance of the federated query engine.

> RQ4: How can we define update interests and propagate interesting updates for manage (co)evolution of data sources?

To answer this research question, we study techniques for update propagation from original data source to its replica, i.e., target data source. We study different cases of update propagation from source to target, and vice-versa, and investigate potential problem that might arise during synchronization when the source and the target are allowed to evolve at the same time. We study the effects of different synchronization strategies on three data quality metrics the completeness, consistency, and conciseness of datasets.

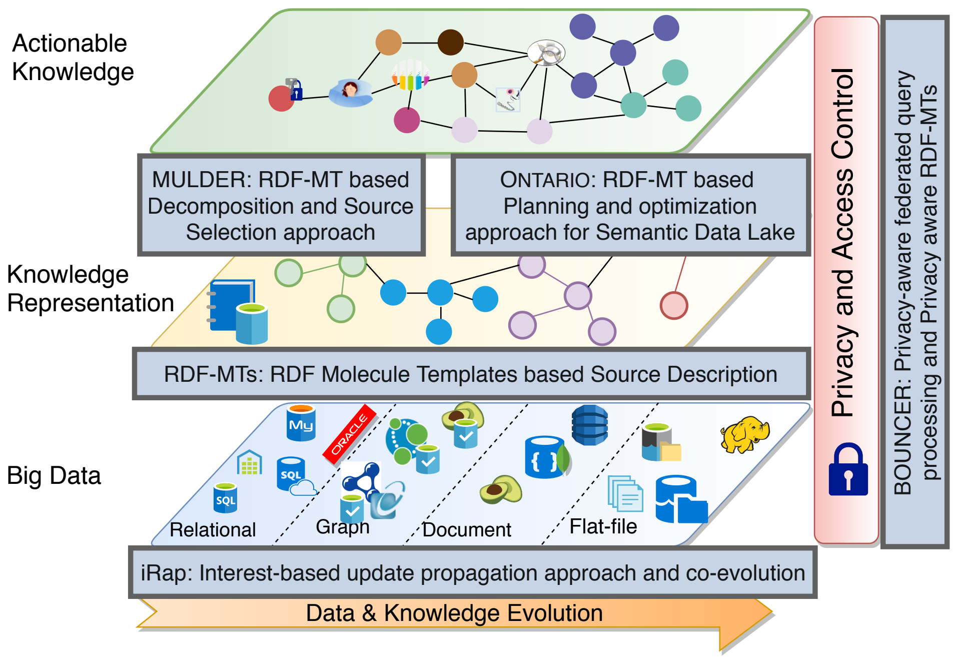

Actionable Knowledge
MULDER: RDF-MT based Decomposition and Source Selection approach
ONTARIO: RDF-MT based Planning and optimization approach for Semantic Data Lake
Knowledge Representation
RDF-MTs: RDF Molecule Templates based Source Description
Big Data
Relational
Graph
Document
Flat-file
iRap: Interest-based update propagation approach and co-evolution
Data & Knowledge Evolution
Privacy and Access Control
BOUNCER: Privacy-aware federated query processing and Privacy aware RDF-MTs

Figure 1.3: **Thesis Contributions:** Four main contributions of this thesis including: (1) RDF Molecule Template (RDF-MT) based source description model, (2) Federated query processing approach over heterogeneous data sources, specifically the MULDER decomposition and source selection and the Ontario planning and optimization approaches, (3) Privacy-aware federated query processing technique, and (4) interest-based update propagation and co-evolution approach

1.4 Thesis Overview

In this section, we present an overview of our main contributions on the research problems investigated by this thesis and related scientific publications.

1.4.1 Contributions

Figure 1.3 shows the main contributions of this thesis.

- **Contribution 1:** *RDF Molecule Template (RDF-MT) based Source Descriptions.* To describe heterogeneous data sources, we propose RDF Molecule Templates (RDF-MTs), an abstract description of entities in a unified schema and their implementation in the federation of data sources. RDF-MTs describe a set of entities that belong to same semantic concept and the relationships between them, i.e., within a data source and between different data sources. In other words, they are templates that represent a set of RDF molecules that share the same semantic concept. RDF-MTs provide a way to analyze the properties of a single data source and set of data sources in a federation, which provide an insight on how dense or sparse the connection of data elements appear in those data sources and the federation as a whole. We demonstrate the application of RDF-MTs for describing data sources and perform high level analysis. In addition, we compare the use of RDF-MTs based query decomposition and optimization with source descriptions computed based on different graph partitioning techniques. The observed results show that RDF-MT based source descriptions capture the semantics encoded in heterogeneous data sources and improved the performance of federated query engine, answering research question **RQ1**.

- **Contribution 2:** *Query processing over heterogeneous data sources.* We address the problem of query processing in two parts. First we tackle the problem of query decomposition and source selection in autonomous, and distributed data sources. We assume these data sources provide a uniform access interface, i.e., SPARQL endpoints via SPARQL query language. Based on this assumption, our optimization techniques focus on the properties of the subqueries and the behavior of data sources during answer generation. We devise MULDER, a federated query processing engine that exploits the RDF Molecule Template (RDF-MT) based data source description of SPARQL endpoints. MULDER provides a novel techniques for query decomposition, source selection, and planning. Then, we generalize these techniques and tackle the problem of plan generation and optimization for autonomous, heterogeneous, and distributed data sources. In this case, data sources might have heterogeneous data model, access interface, and query capabilities. We propose an optimization technique, the Ontario approach, that considers the capabilities of data sources, described by RDF-MTs, during source selection, decomposition, and planning. Ontario employs a greedy algorithm that generates an efficient plan considering a set of heuristics defined based on the characteristics of the query, such as set of star-shaped subqueries, as well as the capabilities of the selected data sources. Evaluation results of MULDER and Ontario shows that query decomposition, source selection, and planning techniques that exploit RDF-MT based source description leads to maximizing answer completeness as well as minimizing execution time, answering research question **RQ2**.

- **Contribution 3:** *Privacy-aware query processing over a federation of data sources* For data sources that contain sensitive data, data providers specify privacy and access control regulations at different level of granularity. To describe privacy and access regulations imposed by data providers we propose privacy-aware RDF-MTs that define privacy and access control policies at the level of predicates, giving the provider a flexibility to restrict at a smallest data element level. We propose BOUNCER, a privacy-aware federated query processing approach that produces a valid plan respecting the privacy and access regulations set by the data provider. BOUNCER is able to select data sources that are relevant to answer the subqueries and is able to eliminate data sources that have restrictions on fragments of data at early stage of query processing. In addition, BOUNCER is able to select operators (algorithms) that respect the privacy and access control policies and generate an execution plan that minimize execution time. The observed results show that a privacy-aware RDF-MTs can encode the privacy and access control policies that are exploited by the privacy-aware federated query engine, BOUNCER, for source selection, query decomposition, and valid plan generation, answering research question **RQ3**.

- **Contribution 4:** *Interest-based update propagation* We propose an interest-based update propagation technique, when access to data locally by replicating or slicing of an evolving data source is preferred instead of direct access from the data provider, to avoid query restrictions set by the provider for performance reasons. Interesting updates are expressed in terms of SPARQL query expressions encoded in RDF-MTs. A target data source, i.e., replica or slice of an evolving data source, is described by RDF-MTs, an abstract description of entities that are replicated by the target data source, and have an interest expressions associated to them. We introduce iRap, an interest-based data propagation technique, that propagates only important changes that are of interest to the target data source. iRap is able to propagate updates in a reasonable time without losing partial updates by keeping potentially interesting fragments locally. In addition, we introduce different synchronization and conflict resolution strategies when both source and replicas are allowed to co-evolve. Experimental results suggest that our interest expression technique capture the data required by the target data source and reduce the amount of data required by several order of magnitude, and our co-evolution techniques positively affect the quality of both data sources, answering research question **RQ4**.

1.4.2 List of Publications

The entire list of publications produced during the PhD study can be found in Appendix A A.1. The work in this thesis is based on the following publications:

1. **Kemele M. Endris**, Sidra Faisal, Fabrizio Orlandi, Sören Auer, Simon Scerri, *Interest-based RDF update propagation*, In Proceedings of the 14th International Conference on The Semantic Web-ISWC 2015-Volume 9366, pp. 513-529. Springer-Verlag, 2015.

2. **Kemele M. Endris**, Sidra Faisal, Fabrizio Orlandi, Sören Auer, Simon Scerri, *iRap-an Interest-Based RDF Update Propagation Framework.* Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.

3. Sidra Faisal, **Kemele M. Endris**, Saeedeh Shekarpour, Sören Auer, Maria-Esther Vidal, *Co-evolution of RDF Datasets*, Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, pp. 225–243, 2016. This is a joint work with Sidra Faisal, an PhD student at the University of Bonn. In this paper, my contributions includes preparing motivating example, contribution to the problem definition and formalization, and preparing datasets for experiments.

4. **Kemele M. Endris**, Mikhail Galkin, Ioanna Lytra, Mohamed Nadjib Mami, Maria-Esther Vidal, Sören Auer, *MULDER: Querying the Linked Data Web by Bridging RDF Molecule Templates.* In International Conference on Database and Expert Systems Applications, pp. 3-18. Springer, Cham, 2017. (**Best Paper Award**)

5. **Kemele M. Endris**, Mikhail Galkin, Ioanna Lytra, Mohamed Nadjib Mami, Maria-Esther Vidal, Sören Auer, *Querying Interlinked Data by Bridging RDF Molecule Templates.* In Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIX, pp. 1-42. Springer, Berlin, Heidelberg, 2018.

6. **Kemele M Endris**, Maria-Esther Vidal, Sören Auer, *FedSDM: Semantic Data Manager for Federations of RDF Datasets.* In International Conference on Data Integration in the Life Sciences, pp. 85-90. Springer, Cham, 2018.

7. **Kemele M Endris**, Zuhair Almhithawi, Ioanna Lytra, Maria-Esther Vidal, Sören Auer, *BOUNCER: Privacy-Aware Query Processing over Federations of RDF Datasets.* In International Conference on Database and Expert Systems Applications, pp. 69-84. Springer, Cham, 2018.

8. Maria-Esther Vidal, **Kemele M. Endris**, Samaneh Jazashoori, Ahmad Sakor, and Ariam Rivas. *Transforming Heterogeneous Data into Knowledge for Personalized Treatments—A Use Case.* In Datenbank-Spektrum: 1-12, Springer, Cham, 2019.

9. Maria-Esther Vidal, **Kemele M. Endris**, Samaneh Jozashoori, Farah Karim, and Guillermo Palma. *Semantic data integration of big biomedical data for supporting personalised medicine.* In Current Trends in Semantic Web Technologies: Theory and Practice, pp. 25-56. Springer, Cham, 2019.

10. **Kemele M Endris**, Philipp D. Rohde, Maria-Esther Vidal, Sören Auer, *Ontario: Federated Query Processing against Heterogeneous Data Sources in a Semantic Data Lake.* International Conference on Database and Expert Systems Applications, Springer, Cham, 2019.
11. Lucie-Aimée Kaffee, **Kemele M. Endris**, Elena Simperl and Maria-Esther Vidal, *Ranking Knowledge Graphs By Capturing Knowledge about Languages and Labels*, Proceedings of the Knowledge Capture Conference (K-CAP) 2019, Marina del Rey, California, USA, ACM 2019
12. David Chaves-Fraga, **Kemele Endris**, Enrique Iglesias, Oscar Corcho and Maria-Esther Vidal, *What are the Parameters that Affect the Construction of a Knowledge Graph?*, Proceedings of the 18th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE) (in OTM) 2019, Rhodes, Greece, Springer 2019.

1.5 Thesis Structure

The remainder of this thesis is organized as follows: Chapter 2 introduces preliminary concepts and theoretical foundations for the research conducted in this thesis. First, it describes the general data integration problem and different classifications based on three dimensions; autonomy, distribution, and heterogeneity. Then, it introduce the Semantic Web as well as main technologies and standards available to realize its vision. Finally, we discuss the basic sub-problems and components of federated query processing systems. In Chapter 3, we present related works to this thesis. We first present state-of-the-art materialized data integration approaches in the Semantic Web community. Then, we give an overview of federated query processing engines that are specialized for RDF data sources. We dive into each sub-problems of federated query processing, i.e., source description, decomposition, source selection, planning, and optimization, specific approaches in the literature. Then, we describe approaches that consider the heterogeneity of data sources during federated query processing. Next, approaches for privacy and access control representation and enforcement are discussed. Finally, we present related works on update propagation and managing co-evolution of datasets that evolve independently to each other. Chapter 4 presents a novel source description model proposed in the scope of this thesis. We utilize the semantics encoded in the data sources to describe different characteristics of the data available in the datasets. We also analyze three different state-of-the-art benchmarks for federated query processing evaluation, and show the expressiveness of the description model as well as the performance improvements compared to other type of source descriptions. Chapter 5 presents a query decomposition and source selection approach, MULDER, utilizing the RDF-MT based source descriptions, presented in Chapter 4. The MULDER approach takes advantage of the semantic descriptions of each data sources and the interlink between them to perform query decomposition and source selection. We show the results of extensive performance evaluation using three different benchmarks that *MULDER* is able to significantly reduce the overall query execution time as well as increase answer completeness. In addition, MULDER has comparable results with other state-of-the-art adaptive query processing engine in terms of continuous answer generation. Chapter 6 presents query planning and optimization techniques utilizing the capabilities of data sources and their semantic description available in RDF-MTs. We present the planning and optimization approach of ONTARIO, a federated query engine over heterogeneous data sources in the Semantic Data Lake, that is able to utilize RDF-MT based descriptions that maximize answer completeness and minimize query execution time. The results of our evaluation provide an evidence that the optimization technique used by ONTARIO is able to speed up query execution and enhance answer completeness with respect to the state-of-the-art SPARQL federated engines. Chapter 7 presents BOUNCER, a privacy-aware query processing techniques, that is able to consider privacy and access control restriction when the federation contains data sources that contain sensitive and non-sensitive data. BOUNCER provides privacy-aware query decomposition and planning technique to generate a valid execution plan according to privacy and access control policy of sources in the federation. We evaluate the performance of BOUNCER assessing the effectiveness and efficiency of the decomposition and planning techniques. The results show that BOUNCER effectively enforce access control and privacy regulations at different level of granularity without impacting the performance of the query processing. Chapter 8 presents an interest-based update propagation technique between evolving data sources. Update interest expressions are described using a graph pattern and conflict resolution strategy, if the target data source is allowed to evolve at the same time. We present an RDF synchronization and conflict identification and resolution approaches. We implemented the RDF synchronization and conflict resolution technique in the iRap framework and perform a comprehensive evaluation based on the DBpedia Live update. We proposed different conflict identification and resolution techniques. Chapter 9 concludes the thesis with a summary of the main results and contributions to the problem of federated query processing against Semantic Data Lakes and present possible future directions for subsequent research work.

CHAPTER 2

Background

In this chapter, we present basic concepts and theoretical foundations for the research conducted in this thesis. First, in Section 2.1, we define the basic concepts and components of a data integration system, then, we present the vision and basic concepts of the Semantic Web, as well as standards and technologies such as Resource Description Framework (RDF), RDF Schema, SPARQL query language, and rule-based mapping languages in Section 2.2. Finally, we show an overview of federated query processing and basic components of a federated query engine in Section 2.3.

2.1 Data Integration System

An enormous amount of data is being published on the Web [7]. In addition, different data sources are being generated and stored within enterprises as well due to technological advances in data collection, generation, and storage. These data sources are created independently of each other and might belong to different administrative entities. Hence, they have different data representation format as well as access interfaces. Such properties of the data sources hinder the usage of data available in them. Data integration is the process of providing a uniform access to a set of distributed (or decentralised), autonomous, and heterogeneous data sources [3]. The number of data sources a data integration system integrate can range from less than a hand full of sources to Web-scale. Data integration systems provide a global schema (also known as mediated schema) to provide a reconciled view of all data available in different data sources. Mapping between the global schema and source schema should be established to combine data residing in data sources considered in the integration process. Generally, data integration system is formally defined as follows [8]:

Definition 1 (Data Integration System) *A data integration system, $\mathbb{I}$, is defined as a triple $< G, S, M >$, where:*

- *G is the global schema, expressed in a language L_G over an alphabet A_G. The alphabet comprises a symbol for each element of G.*
- *S is the source schema, expressed in a language L_S over an alphabet A_S. The alphabet A_S includes a symbol for each elements of the sources.*
- *M is the mapping between G and S, constituted by a set of assertions of the forms: $q_S \rightarrow q_G$, $q_G \rightarrow q_S$; where q_S and q_G are two queries of the same arity, respectively over the source schema S, and over the global schema G. An assertion specify that the connection between the elements of the global schema and those of the source schema.*

Defining schema mapping is one of the main tasks in a data integration system. Schema mapping is the specification of correspondences between the data at the sources and the global schema. The mappings determine how the queries posed by the user using the global schema be answered by translating to the schema of the sources that stores the data. Two basic approaches for specifying such mappings have been proposed in the literature for data integration systems are *Global-as-View(GAV) [9, 10]* and *Local-as-View(LAV)* [11, 12].

2.1.1 Global-as-View Approach (GAV)

Rules defined using the Global-as-View (GAV) approach define concepts in the global schema as a set of views over the data sources. Using the GAV approach, the mapping rules in M defines the concepts of the schema in the sources, S, with each element in the global schema. This means that, a GAV mapping is a set of assertions of the form [8]: $g \rightarrow q_S$, where $g \in G$ and q_S is query over sources in S. A query posed over the global schema, G, needs to be reformulated by rewriting the query with the views defined in, M. Such rewriting is also known as *query unfolding*; the process of rewriting the query defined over global schema to a query that only refers to the source schema. Conceptually, GAV mappings specify directly how to compute tuples of the global schema relations from tuples in the sources. This characteristics of GAV mappings makes them easier for query unfolding strategy. However, adding and removing sources in the GAV approach may involve updating all the mappings in the global schema, which requires knowledge of all the sources. In this thesis, we define a source description model, RDF Molecule Template - an abstract description of entities that share the same characteristics, based on GAV approach. The global schema is defined as a consolidation of RDF-MTs extracted from each data source in the federation. Rule-based mappings, e.g., RML, are used to define the GAV mappings of heterogeneous data sources. RDF-MTs are merged based on their semantic descriptions defined by the ontology, e.g., in RDFS. In Chapter 4 we presents the proposed data source description model in detail.

2.1.2 Local-as-View Approach (LAV)

Mappings specified using the Local-as-View (LAV) approach describe the data sources as views over the global schema. Using the LAV approach, the mapping rules in M associates a query defined over the global schema, G, to each elements of source schema, S, i.e., a LAV mapping is a set of assertions of the form [8]: $s \rightarrow q_G$, where $s \in S$ and q_G is a query defined over the global schema, G. Adding and removing sources in the LAV approach is easier than GAV approach, as data sources are described independently to each other. In addition, it allows for expressing incomplete information as the global schema represents a database whose tuples are unknown, i.e., the mapping M defined by LAV approach might not contain all the corresponding sources for all the elements in thee global schema, G. As a consequence, query answering in LAV may consist of querying incomplete information, which is computationally more expensive [3].

2.1.3 Classification of Data Integration Systems

Data integration systems can be classified with respect to the following three dimensions: autonomy, distribution, and heterogeneity [13], Figure 2.1. **Autonomy** dimension characterizes the degree to which the integration system allows each data sources in the integration to operate independently. Autonomy of data sources could refer to both in terms of control over the data as well as its distribution. Control over the data can be characterized by their autonomy over their choice of their data model, schema, and evolution. Based on the requirements defined by the provider different data representation model might

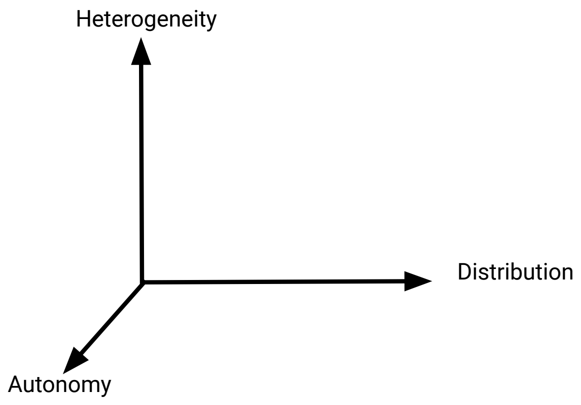


Figure 2.1: Dimensions of Data Integration Systems [13]

be used. Furthermore, sources also have an autonomy to join or leave the integration system at any time as well as to which fragments of data to be accessible by the integration system and its users. **Distribution** dimension specifies the data that is physically distributed across computer networks. Such distribution (or decentralization) can be achieved by controlled distribution or by the autonomous decision of the data providers. Finally, **heterogeneity** may occur due to the fact that an autonomous development of systems yields different solutions, for reasons such as different understanding and modeling of the same real-world concepts, the technical environment, and particular requirements on the application [13]. Though there are different types of heterogeneity of data sources, the important ones with respect to data interoperability are related to data model, semantic, and interface heterogeneity. Data model heterogeneity captures the heterogeneity created by various modeling techniques such that each data models have different expressive power and limitations, e.g., relational tables, property graph, RDF, etc. Semantic heterogeneity concerns the semantic of data and schema in each sources. The semantics of data stored in each source is defined through the explicit definition of their meanings in the schema element under which they are represented. Finally, interface heterogeneity exists if data sources in the integration system are accessible via different query languages, e.g., SQL, Cypher, SPARQL, or API call.

Figure 2.2 shows different classification of data integration systems with respect to distribution and heterogeneity dimensions. The first type of data integration systems, Figure 2.2.(1), loads heterogeneous data from data sources to a centralized storage after transforming them to a common data representation format. The second type of data integration systems, Figure 2.2.(2), support data distributed across networks, however, they only support if the data sources in the system are homogeneous. Data sources are homogeneous in terms of data model and access methods. However, data sources might have different hardware platform, schema, and access restrictions. Federated query processing systems fall in this type when the data sources are autonomous. The third type of data integration systems, Figure 2.2.(3), support data heterogeneity among data sources in the integration system, however, it is managed in centralized way and data is stored in a distributed file system (DFS), such as Hadoop [1]. Finally, the fourth type of data integration systems, Figure 2.2.(4), support data distributed across networks as well as heterogeneity of data sources. Such integration systems utilize special software components to extract data from the data sources using native query language and access mechanism. They can also transform data extracted from the sources to data representation defined by the integration system. Data sources in the integration system might also be autonomous. Federated query processing systems fall in this type if the data sources are autonomous. Such type of systems are different from the third type by how data is distributed and

[1] `https://hadoop.apache.org/`

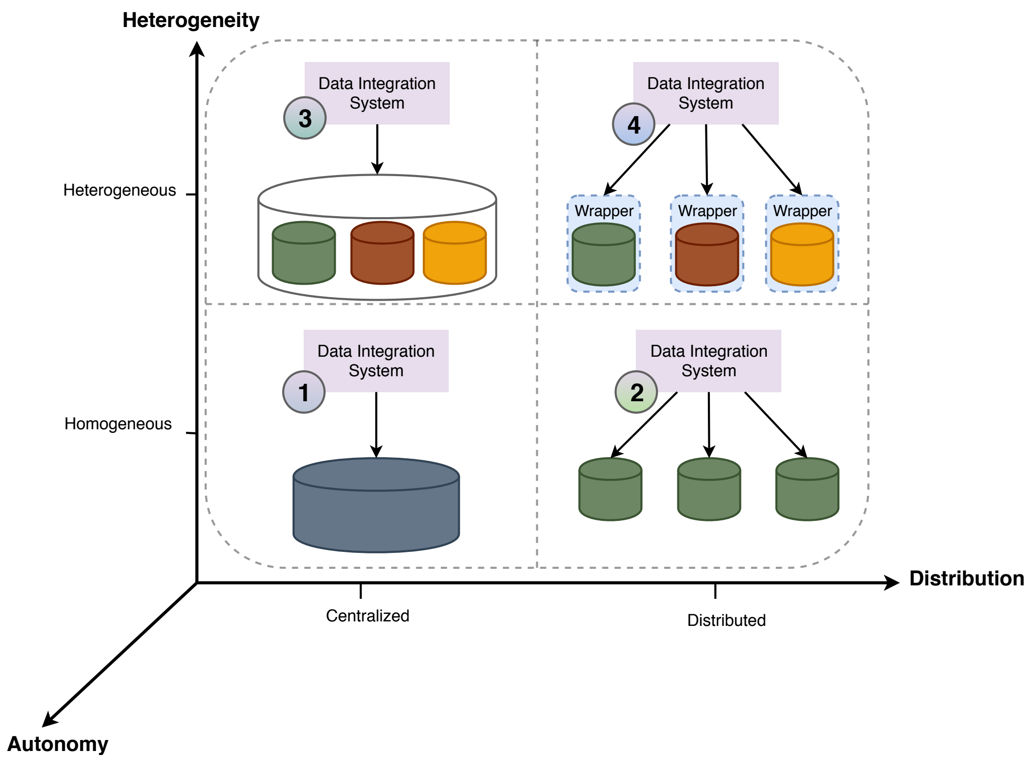


Figure 2.2: Classification of Data Integration Systems

stored. While the fourth type supports any storage management, including DFS, the third type of data integration systems supports only DFS in a centralized way. Mostly the distribution task is handled by the file system. For instance, data might be stored in multi-modal data management system or in a Data Lake storage based only on distributed file system (DFS). In third type of data integration systems, data is loaded from the original source to the centralized storage for further processing.

Data integration systems also have to make sure data that is current (fresh) is accessed and integrated. Especially, for the DFS based Data Lakes, Figure 2.2.(2), and the centralized, Figure 2.2.(4), integration systems, updates of the original data sources should be propagated to guarantee the freshness of data. Furthermore, when accessing original data source from the provider is restricted or management of data in a local replica is preferred, integration systems Figure 2.2.(1) and (3), need to guarantee the freshness of data by propagating changes.

2.1.4 Data Integration in the era of Big Data

In the era of big data, a large amount of structured, semi-structured, and unstructured data is being generated in a faster rate than before. Big data systems that integrate different data sources need to handle such characteristics of data efficiently and effectively. Generally, big data is defined as data whose volume, acquisition speed, data representation, veracity, and potential value overcome the capacity of traditional data management systems [14]. Big data is characterized by a 5Vs model: *Volume* denotes that generation and collection of data are produced at increasingly big scales. *Velocity* represents that data is rapidly and timely generated and collected. *Variety* indicates heterogeneity in data types, formats, structuredness, and data generation scale. Veracity refers to noise and quality issues in the data. Finally, *Value* denotes the benefit and usefulness that can be obtained from processing and mining big data.

There are two data access strategies for data integration: *schema-on-write* and *schema-on-read*. In schema-on-write strategy, data is cleansed, organized, and transformed according to a pre-defined schema before loading to the repository. In schema-on-read strategy, raw data is loaded to the repository as-is and schema is defined only when the data is needed for processing [6]. Data warehouses provide a common schema and require data cleansing, aggregation, and transformation in advance, hence, following the schema-on-write strategy. To provide scalable and flexible data discovery, analysis, and reporting, *Data Lakes* have been proposed. Unlike data warehouses, where data is loaded to the repository after it is transformed to a target schema and data representation, Data Lakes store data in its original format, i.e., schema-on-read strategy. Data Lakes provide a central repository for raw data that is made available to the user immediately and defer any aggregation or transformation tasks to the data analysis phase, thus, addressing the problem of disconnected information silos which is the result of non-integrated heterogeneous data sources in isolated repositories, with diverse schemas and query languages. Such central repository may include different data management systems, such as distributed file systems, relational database management systems, graph data management systems, as well as triple stores for specialized data model and storage, i.e., preserving the rawness of data and constraints represented in it. Data Lakes guarantee eventually a common access interface to available data for processing and analysis task without conveying the development costs of pre-processing and transformations. Along with the raw data, available metadata describing the data sources can also be extracted during the ingestion phase [15]. Metadata governance plays an important role in Data Lakes to efficient discovery of datasets and avoid data swamps. In this thesis, we propose a data source description model, RDF Molecule Template, for Data Lakes, where concepts are described by their shared semantic characteristics. Such descriptions can be used to integrate heterogeneous data sources in the Data Lake based on their semantic description. We present our data description model in Chapter 4.

Semantic integration of big data entails data variety by enabling the resolution of several interoperability conflicts, e.g., structuredness, schematic, representation, completeness, domain, granularity and entity matching conflicts. These conflicts arise because data sources may have different data models, follow various schemes for data representation, contain complementary information [3]. Furthermore, a real-world entity may be represented using diverse properties or at various levels of detail. Thus, data integration techniques able to solve all the interoperability issues while addressing data complexity challenges imposed by big data characteristics are demanded. In this thesis, we focus on the first two dimensions of big data, variety and volume, and resort to a semantic data integration approach that address the challenges imposed by the variety (heterogeneity) of data sources in a Semantic Data Lake. In the next section, we introduce the Semantic Web and available technologies that are defined to realize its vision on solving such interoperability issues.

2.2 Semantic Web

The Semantic Web is an extension of the current Web in which documents on the Web have annotations that make their meanings explicit, making them machine-readable. According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries" [2]. The vision of the Semantic Web is to extend principles of the existing *Web of Document* to *Web of Data*. It provides formalism for representing and accessing data that are translated to a set of standards and technologies used to create data store, vocabularies, and write rules for handling data. At the core of these standards is the Resource Description Framework (RDF) for semantic markup and its associated schema languages, RDF Schema (RDFS) and the Web Ontology Language (OWL). These standards are built on the principles of knowledge representation languages and

[2] https://www.w3.org/2001/sw/

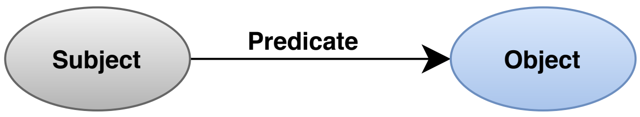


Figure 2.3: **RDF Graph representing an RDF triple.** An RDF triple consisting of a `subject` a resource (represented as a node) to which this triple asserts, a `predicate` as label of a directed edge representing a binary relation, and an `object` which is a value of the predicate that relates it to the `subject` node.

adapt them to the context of the Web, where knowledge is authored in a decentralized fashion by many participants [3]. Linked Data is a set of best practices for publishing and interlinking machine-readable data on the Web [16]. Linked Data enables the consumption of semi-structured data sources on the Web by both human and machines (software agents). To ensure that Linked Data reaches its full potential, datasets should be released under an open license which does not impede its reuse for free. Linked Open Data (LOD) is Linked Data which is released under an open licence [16]. The LOD initiate encouraged data providers to publish a large linked datasets from different domains, which leads to the creation of semantically interconnected global dataspace known as the Linked Open Data Cloud (LOD Cloud) [17]. Prominent datasets in LOD Cloud includes: DBpedia, Wikidata, YAGO, and Bio2RDF. Different domains are represented in the LOD Cloud, including Cross Domain, Geography, Government, Life Science, Linguistics, Social Networking, Media, Publications, and others. During the first release of the LOD Cloud in 2007, there were only 12 datasets interconnected. During the last decade, the LOD Cloud has grown considerably, to the total of $1,239$ datasets with $16,147$ links as of March 2019[3].

2.2.1 The Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a graph-based data model representing information on the Web[4]. The RDF data model allows expressing information in the form of three element tuples, called RDF triples. An RDF triple consists of a `subject`, a `predicate`, and an `object`. A `subject` of an RDF triple denotes a resource or entity that is being described, `predicate` specifies a property or binary relation that associates the subject with the object of the triple, and an `object` of a triple denotes a value of the predicate. A set of RDF triples are called an RDF graph, and a collection of RDF graphs form an RDF dataset. In this thesis, we use the term dataset and data source interchangeably. Nodes in RDF graph can be resources or literals, and RDF resources are identified by IRIs (Internationalized Resource Identifier) or blank nodes (anonymous resources or existential variables). Literals can be enriched with datatypes (defined by XML Schema [18]) and language tags (specified in BCP 47 [19]) in conformance with the RDF specification [20]. RDF resources can be served via native web access interfaces such as dereferencing resource identifiers, and SPARQL endpoint via the SPARQL protocol.

Definition 2 (RDF Triple and Dataset [21]) *Let U, B, L be disjoint infinite sets of IRIs, blank nodes, and literals, respectively. A tuple* $(s, p, o) \in (U \cup B)\ X\ (U)\ X\ (U \cup B \cup L)$ *denotes an RDF triple, where s is called the subject, p the predicate, and o the object. An RDF dataset or RDF graph is a set of RDF triples. When* $s \in L$ *or* $p \in (B \cup L)$*, then the tuple* (s, p, o) *is called a generalized RDF triple and the dataset where it is contained is called a generalized RDF dataset [22].*

[3] https://lod-cloud.net/
[4] `https://www.w3.org/TR/rdf11-concepts/`

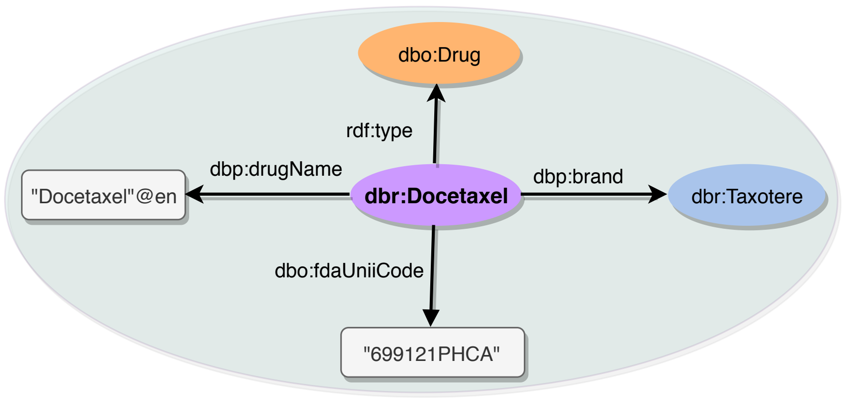


Figure 2.4: RDF Molecule: `dbr:Docetaxel`

Example 1 *The following are examples of RDF triples modeled using DBpedia ontology*[5].

- *Docetaxel is a drug.*
  (`dbr:Docetexel, rdf:type, dbo:Drug`)
- *Docetaxel has name "Docetaxel" in English .*
  (`dbr:Docetaxel, dbp:drugName, "Docetaxel"@en`)
- *Docetaxel is sold under the brand Taxotere.*
  (`dbr:Docetaxel, dbp:brand, dbr:Taxotere`)
- *There are 535,061 inhabitants in Hanover.*
  (`dbr:Hanover, dbo:populationTotal, "535061"^^xsd:integer`)

Graphically we represent RDF graphs as shown in Figure 2.3. An RDF graph is different from any (directed labeled) graph in a way that in an RDF graph a label (predicate) can be used as both edge label as well as a node (subject or object). A set of triples that share same subject value are called *RDF molecules*. Formally, RDF molecules are defined as follows:

Definition 3 (RDF Molecule [23]) *Given an RDF graph G, an RDF molecule $\mathbb{M} \subseteq G$ is a set of triples $\mathbb{M} = \{t_1, t_2, \ldots, t_n\}$ in which $subject(t_1) = subject(t_2) = \cdots = subject(t_n)$.*

Example 2 *The following set of RDF triples represent an RDF molecule for `dbr:Docetaxel`:*

```
dbr:Docetaxel    rdf:type           dbo:Drug   .
dbr:Docetaxel    dbp:drugName       "Docetaxel"@en  .
dbr:Docetaxel    dbp:brand          dbr:Taxotere  .
dbr:Docetaxel    dbo:fdaUniiCode    "699121PHCA"  .
```

Figure 2.4 shows the graphical representation of this molecule.

[5] We assume the following prefixes: `dbo : <http://dbpedia.org/ontology/>`
`dbr : <http://dbpedia.org/resource/>`
`dbp : <http://dbpedia.org/property/>`
`xsd: <http://www.w3.org/2001/XMLSchema#>`
`rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
`rdfs: <http://www.w3.org/2000/01/rdf-schema#>`

2.2.2 RDF Schema

RDF Schema (RDFS) is part of the W3C Recommendation and provides a language to define semantics of user-defined vocabularies for RDF data. RDFS extends RDF with additional modeling constructs that allow to define classes (`rdfs:Class`), hierarchies between classes (`rdfs:subClassOf`) and properties (`rdfs:subPropertyOf`), restrictions on the domains (`rdfs:domain`) and ranges (`rdfs:range`) of properties (`rdf:Property`), and membership (`rdf:type`) of entities in classes. A class represents a set of entities that represent a real-world (or fictional) concept. A membership of an entity to a class is defined via `rdf:type`. Class hierarchies can be specified using the *rdfs:subClassOf* construct. Relationships between classes is defined using the `rdfs:domain` to specify the subject type of a triple and `rdfs:range` to specify the type of object part of a triple defined by a particular property as `rdf:Property`. Hierarchies between properties are specified using `rdfs:subPropertyOf` construct. In addition to these constructs, RDFS provides annotation properties, `rdfs:label` and `rdfs:comment`, to enrich the human-readability of the given resource or entity. The RDF Schema serves as foundation to building an RDF graph with typed hierarchies of concepts and their relationships as defined by using `rdfs:subClassOf` and `rdfs:subPropertyOf`, as well as the restrictions specified using `rdfs:domain` and `rdfs:range`. RDFS also offers a set of entailment rules that are used to infer implicit statements from explicit ones[6].

2.2.3 The SPARQL Query Language and SPARQL Protocol

In 2008, SPARQL[7] query language became a W3C Recommendation for querying RDF data [24]. SPARQL is basically a graph pattern matching query language, as RDF is a directed graph data model. SPARQL queries can be seen having three parts [25]; *pattern matching*, *solution modifiers*, and *output type*. The *pattern matching* part includes several features of pattern matching of graphs, such as optional parts, union parts, nesting, filtering values, and possibility to choose the data source to be matched by the pattern. The *solution modifiers* part allows to modify the values computed by the pattern matching part by applying operators such as projection, distinct, group, order, and limit. Finally , the *output type* part can be yes/no, selections of values of the variables matching the patterns, construction of new RDF data from these values, and descriptions of resources.

A SPARQL query is the form *head*$\leftarrow$*body*, where the *body* of the query is a complex RDF graph pattern expression that may include RDF triples with variables (i.e., triple patterns), conjunctions, optional parts, and constraints over the values of the variables. The *head* part of the query is an expression that indicates how to construct the answer to the query. The evaluation of a SPARQL query against an RDF graph is done in two steps. First, the *body* of the query is matched against the RDF graph to obtain a set of bindings for the variables in the body, then using the information on the *head* of the query, these bindings are processed applying classical relational operators to produce the answer to the query.

The SPARQL language considers operators, `OPTIONAL`, `UNION`, `FILTER`, and `AND` via a point symbol (.), to construct graph pattern expressions. The syntax of SPARQL graph pattern is defined as:

Definition 4 (SPARQL Graph Pattern Expression [25]) *Let V be a set of variables disjoint from $U \cup B \cup L$. A SPARQL graph pattern expression is defined recursively as follows:*

1. *A triple pattern $t \in (U \cup B \cup V) \times (U \cup V) \times (U \cup B \cup L \cup V)$ is a graph expression,*
2. *If P_1 and P_2 are graph patterns, then expressions $(P_1$ AND $P_2)$, $(P_1$ OPT $P_2)$, and $(P_1$ UNION $P_2)$ are graph patterns,*

[6] https://www.w3.org/TR/rdf11-mt/#rdfs-entailment

[7] SPARQL is a recursive acronym that stands for *The SPARQL Protocol and RDF Query Language*

3. *If P is a graph pattern and R is a SPARQL built-in filter condition, then the expression $(P$ FILTER $R)$ is a graph pattern.*

Example 3 *The following expression represents a SPARQL graph pattern composed of a set of triple patterns, AND (.), FILTER, and OPT operators.*

```
{
?drug            rdf:type              dbo:Drug  .
?drug            dbp:drugName          ?name  .
OPTIONAL {
        ?drug    dbp:brand             ?brand  .
        ?drug    dbo:fdaUniiCode      ?fdaUniiCode  .
       }
FILTER  (lang(?name)  =  'EN')
}
```

The SPARQL query language provides four query forms: `ASK`, `SELECT`, `CONSTRUCT`, and `DESCRIBE`. In this work, we focus on SPARQL SELECT queries formally defined as follows:

Definition 5 (SPARQL Expression (SELECT Query) [26]) *Let V be a set of variables disjoint from $U \cup B \cup L$, Q be a SPARQL expression and $S \subset V$ a finite set of variables. A SPARQL SELECT query is an expression of the form $SELECT_S(Q)$.*

Example 4 *The following query represents a SPARQL SELECT query composed of a triple pattern, AND, FILTER, and OPT operators, that projects the first 1000 values that mapped to all variables, i.e., ?drug ?name ?brand ?fdaUniiCode, in the graph pattern expression, i.e., in the body of the query.*

```
PREFIX  dbo  :  <http://dbpedia.org/ontology/>
PREFIX  dbp  :  <http://dbpedia.org/property/>
PREFIX  rdf  :  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
SELECT  ?drug  ?name  ?brand  ?fdaUniiCode
WHERE  {
        ?drug            rdf:type              dbo:Drug  .
        ?drug            dbp:drugName          ?name  .
        OPTIONAL  {
                ?drug    dbp:brand             ?brand  .
                ?drug    dbo:fdaUniiCode       ?fdaUniiCode  .
               }
        FILTER  (lang(?name)  =  'EN')
        }  LIMIT  1000
```

SPARQL SELECT queries are evaluated over an RDF dataset based on mappings, where each mapping represents a possible answer of a query. A *mapping*, μ, is a partial function $\mu : V \rightarrow (U \times B \times L)$, i.e., from a binding from a set of variables to RDF terms. The domain of μ, dom(μ), is the subset of V where μ is defined. Two mappings μ_1 and μ_2 are *compatible*, denoted as $\mu_1 \sim \mu_2$, when $\mu_1 \cup \mu_2$ is also a mapping, i.e., $\forall x \in dom(\mu_1) \cup dom(\mu_2), \mu_1(x) = \mu_2(x)$. Abusing notation, for a triple pattern t, the triple obtained by replacing the variables in t according to mapping μ is denoted by $\mu(t)$ [26]. The semantic of SPARQL graph pattern expression is defined as a function $[[.]]_D$ which translates pattern expression into algebraic operations and returns a set of mappings, formally defined as:

Definition 6 (SPARQL Set Semantics [25, 26]) *Let D be an RDF dataset, t a triple pattern, Q, Q_1, and Q_2 SPARQL expressions, R a filter condition, and $S \subset V$ a set of variables. Let $[[.]]_D$ be a function that translates SPARQL expressions into SPARQL algebraic operations as follows:*

$$\begin{aligned}
[[t]]_D &= \{\mu \mid dom(\mu) = vars(t) \wedge \mu(t) \in D\} \\
[[Q_1\ AND\ Q_2]]_D &= [[Q_1]]_D \bowtie [[Q_2]]_D \\
[[Q_1\ OPT\ Q_2]]_D &= [[Q_1]]_D ⟕ [[Q_2]]_D \\
[[Q_1\ UNION\ Q_2]]_D &= [[Q_1]]_D \cup [[Q_2]]_D \\
[[Q\ FILTER\ R]]_D &= \sigma_R([[Q]]_D) \\
[[SELECT_S(Q)]]_D &= \pi_S([[Q]]_D)
\end{aligned}$$

The semantics of SPARQL query evaluation defined by help of a compact algebra over such mapping sets:

Definition 7 (SPARQL Set Algebra [25, 26]) *Let Ω, Ω_1, and Ω_2 be a set of mappings, R denote a filter condition, and $S \subset V$ be a finite set of variables. SPARQL algebraic operations join ($\bowtie$), union ($\cup$), minus ($\setminus$), left outer join (⟕), projection (π), and selection (σ) are defined as:*

$$\begin{aligned}
\Omega_1 \bowtie \Omega_2 &= \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_2, \mu_2 \in \Omega_2 \colon \mu_1 \sim \mu_2\} \\
\Omega_1 \cup \Omega_2 &= \{\mu \mid \mu \in \Omega_1\ or\ \mu \in \Omega_2\} \\
\Omega_1 \setminus \Omega_2 &= \{\mu_1 \in \Omega_1 \mid for\ all\ \mu_2 \in \Omega_2 \colon \mu_1 \nsim \mu_2\} \\
\Omega_1 ⟕ \Omega_2 &= (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2) \\
\pi_S(\Omega) &= \{\mu_1 \mid \exists \mu_2 \colon \mu_1 \cup \mu_2 \in \Omega \wedge dom(\mu_1) \subseteq S \wedge dom(\mu_2) \cap S = \emptyset\} \\
\sigma_R(\Omega) &= \{\mu \in \Omega \mid \mu \models R\}
\end{aligned}$$

where $\mu \models R$ iff μ satisfies SPARQL built-in filter condition R.

SPARQL basic graph pattern (BGP) is a set of triple patterns and filter patterns that are in conjunctive graph pattern. Formally, BGPs are defines as follows:

Definition 8 (BGP) *Let **U** be the set of all IRIs, **B** be the set of blank nodes, **L** be the set of literals and V be the set of variables. A SPARQL basic graph pattern (BGP) expression is defined recursively as follows:*

1. *A triple pattern $t \in (U \cup B \cup V) \times (U \cup V) \times (U \cup B \cup L \cup V)$ is a BGP;*
2. *The expression (P1 AND P2) is a BGP, where P1 and P2 are BGPs;*
3. *The expression (P FILTER R) is a BGP, where P is a BGP and R is a SPARQL filter expression that evaluates to Boolean value.*

A BGP in a SPARQL query contains at least one star-shaped subquery (SSQ). An SSQ is a non-empty set of triple patterns that share the same subject variable (constant).

Definition 9 (Star-shaped Subquery (SSQ) [27]) *A star-shaped subquery* `star(S, ?X)` *on a variable (constant)* `X` *is defined as:*

1. `star(S,X)` *is a triple pattern* `t={X p o}`*, and* `p` *and* `o` *are different to* `X`*.*
2. `star(S,X)` *is the union of two stars,* `star(S1,X)` *and* `star(S2,X)`*, where triple patterns in* `S1` *and* `S2` *only share the variable (constant)* `X`*.*

Example 5 *The following SPARQL SELECT query is composed of a basic graph pattern (BGP) (composed of five triple patterns, hence four AND operators) and two star-shaped subquery (SSQ) (over subject variables `?drug` and `?brand`, respectively).*

```
PREFIX dbo  : <http://dbpedia.org/ontology/>
PREFIX dbp  : <http://dbpedia.org/property/>
PREFIX rdf  : <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
PREFIX rdfs : <http://www.w3.org/2000/01/rdf-schema\#>
SELECT DISTINCT *
WHERE {
      ?drug         rdf:type          dbo:Drug .
      ?drug         dbp:drugName      ?name .
      ?drug         dbp:brand         ?brand .
      ?brand        rdfs:label        ?brandName .
      ?brand        dbp:price         ?drugprice .
      FILTER ( lang(?name) = 'EN' && lang(?brandName) = "EN" )
      }
```

Complexity of SPARQL

The problem of evaluating SPARQL graph patterns can be considered as a decision problem. The evaluation problem of SPARQL graph patterns is defined as [21]:

INPUT: An RDF dataset D, a graph pattern P, and a mapping μ.
QUESTION: Is $\mu \in [[P]]_D$

The complexity of evaluating SPARQL query is influenced by the type operators, i.e., AND, FILTER, UNION, and OPTIONAL, used in the graph pattern. If the graph pattern expression P is constructed by using only `AND` (and optionally `FILTER`) operator, then the evaluation can be solved in $PTIME$, i.e., $O(|P|.|D|)$ time. Similarly, a graph pattern expression, P, constructed using *only* `UNION` operator (and optionally `FILTER`) the evaluation problem is in $PTIME$. However, if the graph pattern, P, combines *only* the above operators, i.e., `AND`, `UNION`, optionally `FILTER`, then the evaluation problem becomes NP-complete. Finally, the evaluation problem of SPARQL queries constructed with OPTIONAL in combination with any of the above operators becomes *PSPACE*-complete [26].

2.2.4 Rule-based Mapping Languages for Transforming raw data to RDF

Mapping languages defined by the Semantic Web community can be used to transform non-RDF data source to RDF. The rules represent the GAV mappings that define the concepts of an ontology as a set of views over heterogeneous data sources. Such transformation can also be used to transform legacy data bases as well as semi-structured data sources published on the Web. R2RML and RML are exemplar rule-based languages that are widely used for these tasks. Next, we introduce these two mapping languages in detail.

R2RML – RDB to RDF Mapping Language

R2RML is a W3C Recommendation [28] for transformation of relational databases to RDF. R2RML is a language for expressing customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and

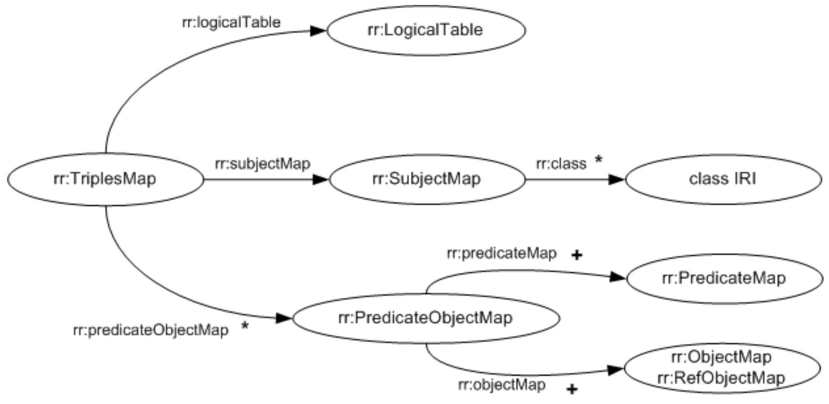


Figure 2.5: Overview of R2RML Triple Maps

target vocabulary of the mapping author's choice. An R2RML mapping is represented as a Triple Map, a rule that maps each row in the logical table to a number of RDF triple. A Triple Map have the following parts (illustrated[8] in Figure 2.5):

- A **Logical Table** (`rr:logicalTable`) refers to a base table, a view, or a valid SQL query to retrieve from the input database. It comprises the following components:
  - **SQL Base Table or View** (`rr:tableName`) specifies the table or view name of the base table or view.
  - **R2RML View** (`rr:sqlQuery` and `rr:sqlVersion`) specifies an SQL SELECT query to be executed against the input database.
- A **Subject map** (`rr:subjectMap`) defines the subjects of the generated RDF triples. Subjects can be defined as IRIs or Blank Nodes. Zero or more class type of subjects can also be defined.
- Zero or More **Predicate-Objece Maps** (`rr:predicateObjectMap`) that in turn consists of:
  - **Predicate Maps** (`rr:predicateMap`) indicates the predicate of the RDF triple and should be a valid IRI;
  - **Object Maps** (`rr:objectMap`) indicate the object of the RDF triple;
  - **Referencing Object Map** (`rr:refObjectMap`) that indicates the reference to another Triple Map, if not specified as **Object Map**.

RML – RDF Mapping Language

RDF Mapping Language (RML) extends R2RML by generalizing to heterogeneous data sources. RML is a generic mapping language defined for expressing customized mappings from heterogeneous data sources, e.g., RDB, CSV, XML, JSON, to the RDF data model. Each mapping rule in RML is represented as a Triple Map which consists of the following parts[9], illustrated [10] in Figure 2.6;

- A **Logical Source** (`rr:logicalSource`) that refers to a data source from where data is collected; it is composed of the following components:

[8] `https://www.w3.org/TR/r2rml/images/triples-map.png`
[9] `http://rml.io/RMLmappingLanguage.html`
[10] `http://rml.io/img/RML_R2RML.png`

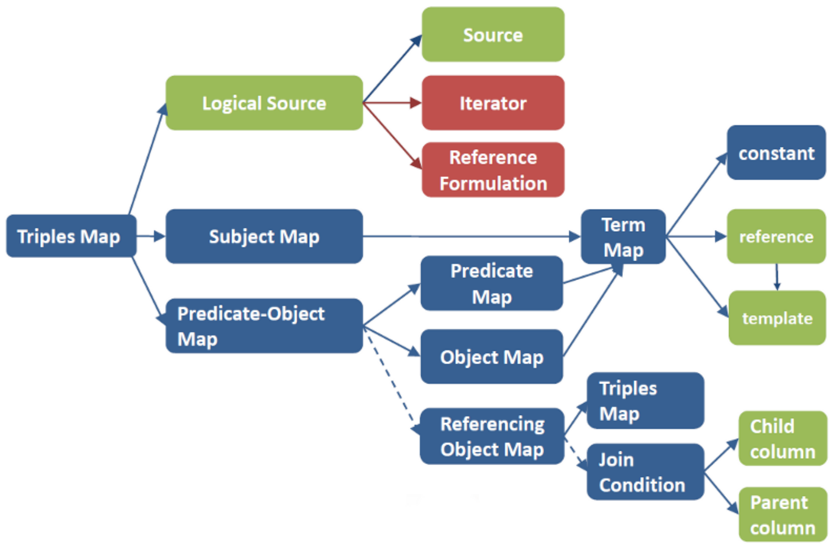


Figure 2.6: RML mapping schema

  - **Source** (`rml:source`) - is an input source, can be JSON, XML, CSV, or a data management system such as RDB, NoSQL store;
  - **Iterator** (`rml:iterator`) - is not required when it comes to tabular input sources, like relational databases. It is needed in case of hierarchical or structured data sources. The iterator (`rml:iterator`) determines the iteration pattern by the input source and specifies the extraction of the data displayed during each iteration;
  - **Reference Formulation** (`rr:referenceFormulation`) - as RML deals with different data serializations with various ways to refer to their elements, RML defines the reference. Such reference is specified based on the source of the input data file, e.g. in case of a JSON file, a Reference Formulation would be "JSONPath", in case of an XML file, a Reference Formulation would be "XPath".

- A **Subject Map** (`rr:subjectMap`) - defines the subject of the generated RDF triples.
- Zero or more **Predicate-Object Maps** (`rr:predicateObjectMap`), combining:
  - **Predicate Maps** (`rr:predicate`) expressing the predicate of the RDF triple and must be a valid IRIs;
  - **Object Maps** (`rr:objectMap`) expressing the object of the RDF triple and must be either IRIs, Blank Nodes or Literals;
  - A **Referencing Object Map**, that indicates the reference to another (existing) Triples Maps (`rr:parentTriplesMap`) and can have zero or more join conditions.

Each **Predicate-Object Map** should have at least one **Predicate Map** and one either **Object Map** or **Referencing Object Map**.

Listing 2.1: Example RML Mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix ql: <http://semweb.mmlab.be/ns/ql#>.
@prefix vocab: <http://example.com/vocab/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

<#AirportMapping>
  rml:logicalSource [
    rml:source "http://www.example.com/drugs.csv" ;
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template "http://example.com/Drug/{name}"; rr:class vocab:Drug  ];
  rr:predicateObjectMap [
      rr:predicate rdfs:label;  rr:objectMap [  rml:reference "Name" ]
    ];
  rr:predicateObjectMap [
    rr:predicate vocab:chemicalFormula;
    rr:objectMap [
      rml:reference "Formula"; rr:datatype xsd:string ]
  ];
  rr:predicateObjectMap [
    rr:predicate vocab:avgWeight;
    rr:objectMap [
      rml:reference "Avg Weight"; rr:datatype xsd:decimal ]
  ].
```

Table 2.1: Example *drugs.csv*

Name,	**Formula**,	**Avg Weight**
Docetaxel,	$C_{43}H_{53}NO_{14}$,	807.8792

Listing 2.2: Example RDF Output

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix vocab: <http://example.com/vocab/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix drug: <http://example.com/Drug/> .
drug:Docetaxel rdf:type              vocab:Drug;
               rdfs:label            "Docetaxel";
               vocab:chemicalFormula "C43H53NO14"^^xsd:string;
               vocab:avgWeight       "807.8792"^^xsd:decimal.
```

2.3 Federated Query Processing Systems

A federated query processing system[11] provides a unified access interface to a set of autonomous, distributed, and heterogeneous data sources. While distributed query processing systems have control over each dataset, federated query processing engines have no control over datasets in the federation and data providers can join or leave the federation at any time and modify their datasets independently. Query processing in the context of data sources in a federation is difficult than centralized systems, because of the different parameters involved that affect the performance of the query processing engine [3]. Data sources in a federation might contain fragments of data about an entity, have different processing capabilities, support different access patterns, access methods, and operators. The role of federated query processing engine is to transform a query expressed in terms of the global schema, i.e., the federated query, into an equivalent query expressed in the schema of the data sources, i.e., local query. The local query represent the actual execution plan of the federated query by the data sources of the federation. The transformation of the federated query to local query need to be both effective and efficient. It is effective if the transformed query has the same semantics and able to produce same results as the

[11] In this thesis, we use *federated query processing engine*,*federated query engine*, and *federated query processing system* interchangeably

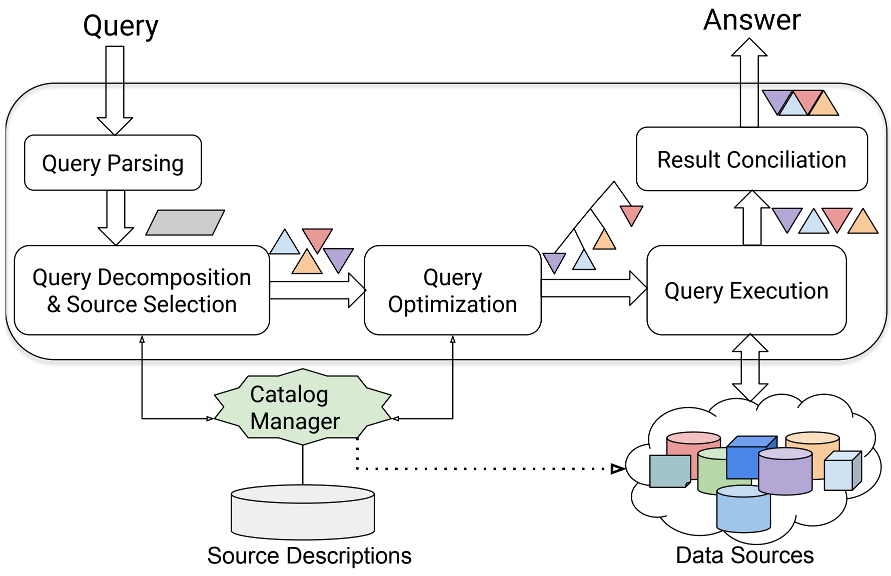


Figure 2.7: Federated Query Processing Basic Components

federated query. On the other hand, it is efficient if the execution strategy of the transformed query use minimum computational resources and communication cost. Producing an efficient execution strategy is difficult, since many equivalent and correct transformations can be produced and each equivalent execution strategy leads to different consumption of resources [13].

The main objective of query processing in the federated context is to transform the federated query posed on the federation of data sources, seen as a single data source by the users, into an efficient execution plan expressed in schema of the data sources and source query language, if different from language [13] used by the federated query. An important part of query processing in the context of federated data sources is query optimization, since many execution plans are correct transformations of the same federated query, the one that optimize (minimize) resource consumption should be retained. The performance of query processors can be measured by the total cost that will be used in processing the query and the response time of the query, i.e., the time elapsed for executing the query.

As an RDF data model continues gaining popularity, publicly available RDF datasets are growing in numbers and size. One of the challenges emerging from this trend is how to efficiently and effectively execute queries over a set of autonomous RDF datasets. Saleem et al. [29] study federated RDF query engines with Web access interfaces. Based on their survey results, the authors divide federation approaches into three main categories: *Query Federation over SPARQL endpoints*, *Query Federation over Linked Data (via URI lookups)*, and *Query Federation on top of Distributed Hash Tables*. Moreover, Acosta et. al [30] classified federated RDF query processing engines based on the type of data sources they support into three categories: *Federation of SPARQL endpoints*, *Federation of RDF Documents*, and *Federation of Triple Pattern Fragments*.

Conceptually, federated query processing involves four main sub-problems (components): (i) *data source description*, (ii) *query decomposition and source selection*, (iii) *query planning and optimization*, and (iv) *query execution*. Federated query engines also include two additional sub-problems: *query parsing* and *result conciliation*. Query parsing and result conciliation sub-problems deals with syntactic issues of the given query and formatting the results returned from the query execution, respectively. Below we provide an overview of the data source description, query decomposition and source selection, query planning and optimization as well as query execution sub-problems.

2.3.1 Data Source Description

Data source description sub-problem deals with describing the data available in data sources and managing catalog about data sources that are participating in the federation. Data source descriptions encode information about available data sources in the federation, types of data in each data source, access method of data sources, and privacy and access policies of these data sources [3]. The specification of what data exist in data sources and how the terms used in data sources are related to the global schema are specified by the schema mapping. Schema mappings also represent privacy and access control restrictions as well as statistics on the available data in each data sources. Federated query engines rely on the description of data sources in the federation to select relevant sources that may contribute to answer a query. Data source descriptions are utilized by the source selection, query decomposition, and query optimization sub-problems.

Catalog of data source descriptions can be collected offline or during query running-time. Based on the employed catalog of source descriptions, SPARQL federation approaches can be divided into three categories [29]: *pre-computed catalog assisted, on-the-fly catalog assisted, and hybrid (uses both pre-computed and on-the-fly)* solutions. Most of state-of-the-art pre-computed catalog assisted federated SPARQL query engines uses three types of catalogs: service descriptions, VoID (*Vocabulary of Interlinked Datasets*) description, and list of predicates [31]. The first two catalogs are computed and published by the data source providers that contains descriptions about set of vocabularies used, list of classes and predicates, as well as some statistics about the instances such as number of triples per predicate, class, etc. Specifically in VoID descriptions, information about external linksets that indicate the existance of *owl:sameAs* and other linking properties. The third type of catalog, i.e., list of predicates, are generated by contacting the data source endpoints and issuing SPARQL queries and extracting predicates from the other two type of catalogs.

2.3.2 Query Decomposition and Source Selection

Selecting the relevant data sources for a given query is one of the sub-problems in federated query processing. Given a federated query parsed with no syntactic problems, the query is first checked if it is semantically correct with respect to the global schema. This step eliminates an incorrect query that yields no results early on. The query is then simplified by, for example, removing redundant predicates. The task of source selection is to select the actual implementation of subqueries in the federation at specific data sources. The sources schema and global schema are given by the data source descriptions as input to this sub-problem. The query decomposition and source selection sub-problem decomposes the federated query into subqueries associated with data sources in the federation that are selected for executing the subqueries. The number of data sources considered for selection are bounded by the data source description given to the federated query processing engine. Each sub-query may be associated to zero or more data source, thus, if the query contains at least one sub-query with out data source(s) associated with it, then the global query can be rejected. Source selection task is a critical part of query optimization. Failure on selecting correct data sources might lead to incomplete answers as well as high response time and resource consumption. The output of this component is a decomposed query into subqueries that are associated with the selected data sources in the federation. Identifying the relevant sources of a query not only leads to a complete answer but also faster execution time.

2.3.3 Query Planning and Optimization

The goal of query planning is to generate an execution plan that represent the steps on how the query is executed and which algorithms (operators) are used. The task of query plan generation produces query execution plan, e.g., a tree-based plan where the leaf of the tree corresponds to the sub-queries to be executed in selected data sources and the internal nodes corresponds to the physical (algebraic) operators, such as join, union, project, and filter, that perform algebraic operation by the federated query processing engine. Many semantically equivalent execution plans can be found by permuting the order of operators and subqueries. However, the cost of executing different ordering of a query is not always the same. In a federated setting, the number of intermediate results as well as the communication costs impact the performance of query execution. Federated query processing engines should use an optimization techniques to select an optimal execution plan that reduce the execution time and resource usage, such as memory, communication, etc. Optimization of the query execution plan starts from selecting only relevant sources, decomposition and finally making decision on selection of an appropriate implementation of join operations. These optimization techniques include making decision on selection of the join methods, ordering, and adapting to the condition of the sources. The objective of the planning and optimization sub-problem is to find an execution plan that minimize the cost of processing the given query, i.e., finding the "best" ordering of operators in the query which is close to optimal solution. Finding an optimal solution is computationally intractable [32]. Assuming a simplified cost function, it is proven that the minimization of this cost function for a query with many joins is NP-Complete. To select the ordering of operators it is necessary to estimate execution costs of alternative candidate orderings. There are two type of query optimization in the literature: cost-based and heuristics-based query optimization. In cost-based optimization techniques, estimating the cost of the generated plans, i.e., candidate orderings, requires to collect statistics on each data sources; either before query executions, *static optimization*, or during query execution, *dynamic optimization*. In a federated settings, where data sources are autonomous, collecting such statistics might not always be possible. Cost-based approaches often are not possible because the data source descriptions do not have the needed statistics. Heuristic-based optimization techniques can be used to estimate the execution cost using minimum information collected from sources as well as the properties of the operators in the query, such as type of predicates, operators, etc. The output of the query planning and optimization is an optimized query, i.e., query execution plan, with operations (join, union) between sub-queries.

2.3.4 Query Execution

Query execution is performed by data sources that are involved in answering sub-query(s) of the given query. Each sub-query executed in each data source is then optimized using the local schema and index (if available) of the data source and executed. The physical operator (and algorithms) to perform the relational operators (join, union, filter) may be chosen. Five different join methods are used in federated query engines: nested loop join, bound-join, hash join, symmetric join, and multiple join [31]. In nested-loop join (NLJ) the inner sub-query is execute for every binding of intermediate results from the outer sub-query of the join. The bindings that satisfy the join condition are then be included in the join results. Bound-join, like NLJ, executes inner sub-query for the set of bindings, unlike NLJ which executes the inner sub-query for every single binding, of the intermediate results from the outer sub-query. This set of binding can be sent as a UNION or FILTER SPARQL operators can be used to send multiple bindings to the inner sub-query. In hash-join method each sub-queries (operands of the join operation) are executed in parallel and the join is performed locally using a single hash table at the query engine. The fourth type of join method, symmetric (hash) join, is non-blocking hash-based join that pipelining parallel execution of the operands and generates output of the join operation as early as possible. Several extended versions of this method are available, such as XJoin [33], agjoin [34], and adjoin [34]. Finally, multiple (hash) join method uses multiple hash tables to join more than two sub-queries running at the same time.

CHAPTER 3

Related Work

In this chapter, we review state-of-the-art approaches related to this thesis. Figure 3.1 shows the topics identified for the literature review. For each topic, we present an overview of the approaches and their limitations that are in the scope of the challenges defined by this thesis. Our literature review will focus on approaches that are proposed to solve the interoperability issues during data integration and update propagation exploiting the semantic technologies by the Semantic Web community. Section 3.1 discusses state-of-the-art semantic-based materialized (data warehousing) integration approaches in the literature. Then, Section 3.2 presents a review of approaches in the area of federated query processing techniques focusing on different sub-problems: source description, source selection, query decomposition, query planning, and optimization. In Section 3.3, we discuss existing solutions to the problem of query processing over heterogeneous data sources. Privacy-aware federated query processing techniques in the reviewed literature are discussed in Section 3.4. Finally, Section 3.5 presents existing works in the area of update propagation and co-evolution.

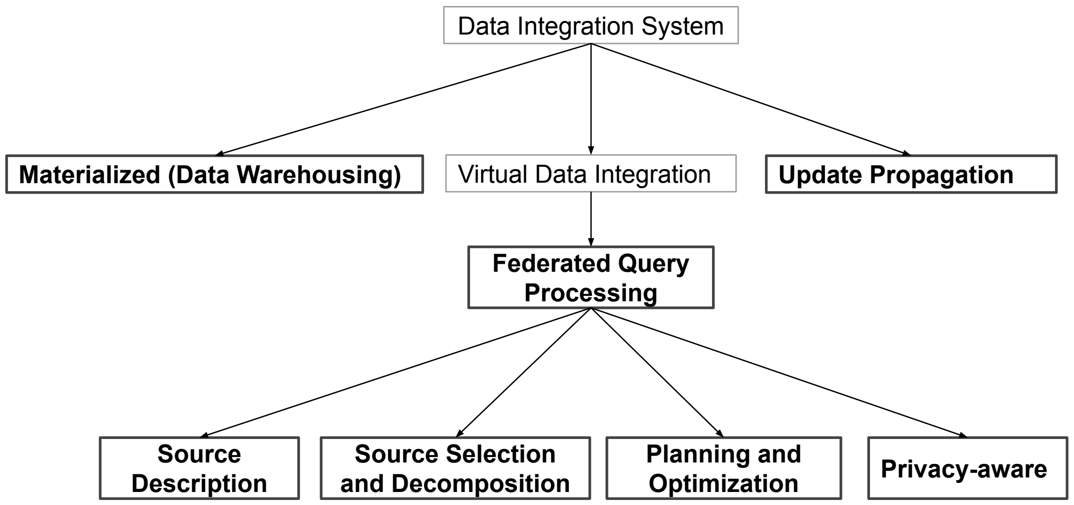

Data Integration System
Materialized (Data Warehousing)
Virtual Data Integration
Update Propagation
Federated Query Processing
Source Description
Source Selection and Decomposition
Planning and Optimization
Privacy-aware

Figure 3.1: Related work topics: We present the works that are related to this thesis in three topics: Materialized (data warehousing), Federated Query Processing, and Update propagation

3.1 Materialized Integration Approaches

Materialized (data warehousing) integration approaches follow schema-on-write paradigm and uses Extract-Transform-Load (ETL) processes to transform and integrate heterogeneous data to a centralized store. These approaches aims at producing a consolidated dataset that allows to query and perform analytic on top of it. Materialized data integration frameworks in the Semantic Web literature includes Linked Data Integration Framework (LDIF) [35], Information Workbench [36], Semantic Web Pipes [37], and ODCleanStore [38] which are used to transform heterogeneous data to RDF graph.

LDIF is a framework that translates heterogeneous data from the Web into a homogenized local target, i.e., RDF data. LDIF maintains an integration pipeline including components for data retrieval, schema mapping, duplicate detection, quality assessment, and fusion. After retrieving data from the Web, using the R2R mapping language [39], the schema translation phase translates the source vocabularies to a local target vocabulary. LDIF uses the SILK [40], identity resolution and linking tool, to discover URI aliases and replace them with a single target URI (also adding `owl:sameAs` links to the original sources). Then, LDIF performs data quality assessment and conflict resolution from different sources using Sieve [41]. Finally, the output of the integration process as accompanied by provenance information and written in a quads format, i.e., triples are extended by a fourth component that carries the provenance information. ODCleanStore [38] supports the management of Linked Data including tasks such as data cleaning, linking, transformation, and quality assessment. As a first step, ODCleanStore carries out data cleaning on each source dataset. ODCleanStore, like LDIF, employs SILK for entity resolution and linking. ODCleanStore uses manually provided trust scores for named graphs and it then computes an aggregated quality score based on the scores of the sources (as named graphs). Once duplicate quads have been identified by SILK, ODCleanStore normalizes their subject URIs into a single URI, removes duplicates and groups quads with conflicts. Then, for each set of conflicting quads, appropriate conflict resolution policies (e.g., MIN, MAX, BEST, AVG) are applied to a value, considering the trust scores of their source graph. The output of the pipeline is a set of integrated quads accompanied by provenance and quality scores of the source. Bischof et al. [42] presented a platform for collecting, integrating, and enriching open data about cities. Their pipeline collects, cleans, and integrates various open data sources; then uses statistical regression methods for predicting and filling in the missing values.

Approaches for materialized data integration produces a consolidated dataset extracted, transformed to RDF and then loaded to a centralized storage, e.g., triple store. However, these approaches could suffer from different issues on different dimensions of big data, such as volume and variety. Integrating a large scale data, such as in biomedical domain, into a centralized data also suffers different performance issues and it might be out of date due to the changes made by the original data sources. The main issues with materialized data integration includes scalability, volatility (freshness of data), and it could take longer time to get the infrastructure running.

3.2 Federated Query Processing Systems

The problem of query processing over federations of data sources has been extensively studied by the database and Semantic Web communities. Existing solutions rely on a global or unified interface that allows for executing queries on a federation of autonomous, distributed, and potentially heterogeneous data sources in a way that execution time is minimized while answer completeness is maximized. Federated query processing engines in the database community employ the relational model to represent the unified view of the federation [43–47], and the query language **SQL** is utilized to express queries against the federation of data sources. On the other hand, federated query processing engines in the

Semantic Web community employ the RDF data model and exploit the semantics encoded in data sources of the federation to build a catalog of data source descriptions [27, 34, 48–51]. Such data source descriptions are used to select the sources from the federation where SPARQL (sub-)queries will be executed. Furthermore, heuristic- and cost-based approaches employ these source descriptions to perform source selection and query decomposition, query planning, and optimization. Existing federated query processing engines from the Semantic Web literature include ANAPSID [34], FedX [52], Avalanche [48], Lusail [53], SPLENDID [51], and Semagrow [49]. Other approaches, like, *Linked Data Fragments* (LDF) [54] provide distributed storage and federated querying element as TPF (Triple Pattern Fragments) client. TPF is optimized for processing *triple patterns*. In what follows, we discuss the state-of-the-art approaches for each federated query processing sub-problems in the literature.

3.2.1 Data Source Description and Source Selection Approaches

Federated query processing engines employ a catalog of data source descriptions for selecting relevant data sources to answer a given query. Such data source descriptions can be extracted at query time or computed beforehand. FedX [52] does not require a catalog of source descriptions computed beforehand, but uses triple pattern-wise ASK queries sent to data sources at query time. Triple pattern-wise ASK queries are SPARQL ASK queries which contain only one triple pattern in the graph expression of the given query. Lusail [53], like FedX, uses a on-the-fly catalog solution for source selection and decomposition. Unlike FedX, Lusail takes an additional step to check if pairs of triple patterns can be evaluated as one subquery over a specific endpoint; this knowledge is exploited by Lusail during query decomposition and optimization. Posting too many SPARQL ASK queries can be a burden for data sources that have limited compute resources. which may result in DoS.

Pre-computed catalog of data source descriptions can be used to reduce the number of requests sent to the data sources. ANAPSID [34] is a federated query processing engine that employs a hybrid solution and collects a list of RDF predicates of the triple patterns that can be answered by the data sources and sends ASK queries when required during query time. During the source selection, ANAPSID parses the SPARQL query into *star-shaped subqueries* and identifies the SPARQL endpoints for each subquery by utilizing predicates list computed beforehand. For triple patterns that are not found in the catalog, ANAPSID sends SPARQL ASK query to data sources if they could be answered by any of the existing sources in the federation. Similarly, HiBISCuS [55], a source selection approach, uses a hybrid solution to collect a catalog of data source descriptions that combines service descriptions computed beforehand with triple-pattern wise ASK queries. The data source description of HiBISCuS includes additional information on the subject and object values of predicates which relies on authority fragment of URIs gathered for each endpoint. HiBISCuS source selection approach discards irrelevant sources for a particular query by modeling SPARQL queries as *hypergraphs*.

Publicly available dataset metadata are utilized by some federated query processing engines as catalog of source descriptions. SPLENDID [51] relies on instance-level metadata available as *Vocabulary of Interlinked Datasets* (VoID) [56] for describing the sources in a federation. SPLENDID provides a hybrid solution by combining VoID descriptions for data source selection along with SPARQL ASK queries submitted to each dataset at run-time for verification. Statistical information for each predicate and types in the dataset are organized as inverted indices, which will be used for data source selection and join order optimization. Similarly, Semagrow [49] implements a hybrid solution, like SPLENDID, and triple pattern-wise source selection method which uses VoID descriptions (if available) and SPARQL ASK queries. *Avalanche* [48] is a federated query engine which also identifies relevant sources and plans the query based on online statistical information published as VoID descriptions. Although VoID allows for the description of a dataset statistics, this description is limited and lacks details necessary for efficient

Approach	**Catalog**	**ASK**	**Privacy-aware**
FedX	x	✓	x
ANAPSID	Predicates list	✓	x
SPLENDID	VoID Desc.	✓	x
Lusail	x	✓	x
Semagrow	VoID Desc.	✓	x
HiBISCuS	Service Desc. + URI auth	✓	x
Avalance	VoID Desc.	x	x
Odyssey	FCP	x	x
DAW	Service Desc. + MIPs	✓	x
DARQ	Service Desc.	x	x
FEDRA	FEDRA index	x	x
SAFE	Data Cubes	✓	✓

Table 3.1: Overview of data source description approaches supported by state-of-the-art federated query processing engines. While some of the federated query processing engines send only ASK queries to check if triple pattern(s) can be executed at query time, most of them uses this method only when a description related to a triple pattern cannot be found in their catalog. Only one federated query processing engine, i.e., SAFE, supports privacy and access control specifications in their description.

query optimization. For instance, though VoID descriptions provide information about link existence between datasets via a linking property, it is not clear in which class(es) this property belongs too. In addition, VoID descriptions could be out-of-date if the dataset updates are very frequent. Odyssey [50] collects detailed statistics information on datasets that enable cost estimation which may lead to low-cost execution plans. The optimization is based on a cost model using statistical methods used for centralized triple stores, i.e., Characteristics Set (CS) [57] and Characteristics Pairs (CP) [57, 58]. Odyssey identifies CSs and sources using predicates of each star-shaped subquery. Then, it prunes to non-relevant sources based on links between star-shaped subqueries and by finding *Federated Characteristics Pairs (FCPs)*. However, unexpected changes and misestimated statistics may conduce to poor query performance.

Different data sources in a federation could contain duplicated data or can be replicas of a dataset. DAW [59] is a duplication-aware hybrid solution for triple pattern wise source selection; it uses the DAW index to identify sources that lead to duplicated results and skip those sources. After making triple pattern-wise source selection, the selected sources are ranked based on the number of new triples they provide; those sources that are below a threshold are skipped. Duplicates are detected using Min-Wise Independent Permutations (MIPs) stored in the DAW index for each triple within the same predicate. FEDRA [60] is a source selection strategy for sources with a high replication degree. FEDRA relies on schema-level fragment definitions and fragment containment to detect replication; it exploits replication information to minimize data redundancy and data transfer by reducing the number of unions, i.e., by minimizing the number of endpoints selected.

In this thesis, we propose RDF Molecule Template based source descriptions that leverage the semantics encoded in data sources. We formally define RDF Molecule Templates (RDF-MTs) and devise techniques for exploiting RDF-MTs during source selection, query decomposition, and planning. Unlike FedX and Lusail, our approach collects RDF Molecule Templates (RDF-MTs) beforehand, reducing the number of requests sent to a data source during query time. Our approach describes sources as a set of RDF-MTs, where each RDF-MT describes a set of RDF molecules that have the same characteristics, such as `rdf:type` values, and possible properties associated to them. An RDF-MT also contains a set of links that exist within the same source and a set of links with other RDF-MTs in different data sources.

Given a SPARQL query, our decomposition and source selection approach parses it into star-shaped subqueries and creates a *query-graph* where nodes are star-shaped subqueries and edges are join variables. Using RDF-MT based source descriptions, for each node in the *query-graph*, our approach selects the RDF-MT(s) that contain all or subset of predicates of a star-shaped subquery. Finally, our source selection approach selects a source for a star-shaped subquery if it is described by a RDF-MT with properties that appear in the triple patterns of the subquery. Once the RDF-MT(s) are selected for the subqueries of a query, information about links between RDF-MT(s) is used to prune the RDF-MT(s) and select only the relevant sources; thus, speeding execution time without impacting query completeness.

3.2.2 Query Decomposition Techniques

Once sources are selected, subqueries are decomposed into a form that will be sent to each selected source. FedX [52] introduces the concept of *Exclusive Groups (EG)* by combining a set of triple patterns that can be sent to the same data source. FedX uses a *triple-pattern based source selection*, where data sources are identified using SPARQL ASK queries for each triple patterns in the given query. Then, a set of triple patterns that matches only to one data source and same source are combined as one sub-query that can be executed once by the selected data source. This reduces the number of requests sent to the data sources as well as pushes down join operations, if there are any, to the data sources. One issue with this approach is that triple patterns are merged regardless of weather they share same variable or not. Executing disjoint patterns, i.e., patterns that do not share any variable, could be expensive as the result is the cross-product of the patterns. Lusail [53] is another federated query engine that presents a locality aware decomposition which uses the concept of *Global Join Variable* and source descriptions to decompose queries, similar to Exclusive Groups in FedX. *Global Join Variable* is a variable that exists at least in two triple patterns for which the solution should come from more than one SPARQL endpoint.

Semagrow [49] follows heuristics that group multiple triple patterns that could be sent to the same source into a single query. These heuristics utilize cardinality estimations of combined triple patterns that reduce the search space of Semagrow optimizer, as it uses cost-based optimizer using dynamic programming (DP). This step have impact on the optimization time to illuminate possible plans that will generate unnecessary intermediate results. Additionally, if data sources are known to mirror another source, then alternative plans are created rather than a single plan. ANAPSID, on the other hand, provides two heuristics for query decomposition [61]: SSGS (Star-Shaped Group Single endpoint selection) and SSGM (Star-Shaped Group Multiple endpoint selection). SSGS reduces the number of unions by selecting only a single source among relevant sources that can answer a star-shaped subquery. That means, SSGS trade answer completeness for faster execution time. On the other hand, SSGM creates a set of UNIONs for each subquery that have more than one selected source. Although SSGS performs better in terms of execution time than SSGM, it could return incomplete results since it only selects at most one source per subquery. In addition, ANAPSID uses Exclusive Groups (EG) technique before applying either SSGS or SSGM decomposition techniques. Unlike FedX, ANAPSID combines triple patterns to be sent to a single source only if they share same variable name, i.e., if they have same join variable either in the subject or object part of triple patterns. Odyssey [50] uses heuristics similar to ANAPSID, and further combines star-shaped subqueries to a single SPARQL query to a particular endpoint whenever the same source for star-shaped subqueries is selected. Such decomposition techniques could be a burden for endpoints with limited resources [54].

The decomposition technique proposed in this thesis utilizes RDF-MTs for both source selection and query decomposition. It combines triple patterns in a star-shaped subquery as a single SPARQL query to be sent to each selected data sources; thus, less source connections are required and query execution is speed up. Especially for Big Data sources, combining not only triple patterns but also star-shaped groups speed up the execution time, by pushing down the processing near the voluminous data sources of the federation. The decomposition technique also adopts to the type of data sources selected and the type of star-shaped groups.

3.2.3 Query Planning and Execution Techniques

Federated query processing engines, such as ANAPSID [34] and FedX [52], follow heuristics-based optimization techniques, while Semagrow [49], SPLENDID [51], Lusail [53], and Odyssey [50] implement cost-based optimization techniques to find a low cost plan. ANAPSID [34] is an adaptive federated query engine capable of delivering query results as soon as they arrive from the data sources, i.e., by implementing non-blocking join operators. On the other hand, FedX [52] is a non-adaptive federated query processing approach, at the level of query execution, that optimizes query execution by introducing exclusive groups, i.e., a set of triple patterns that can be executed against only one data source in the federation. In ANAPSID and FedX, selectivity of subqueries is estimated by counting the ratio of constants over variables in the subquery. The cost estimation of Semagrow [49] is based on a cost model over each operator using either statistics provided by source descriptions or estimated cardinality of sub-expressions. The cost of operators is estimated by applying a communication overhead factor to the cardinality of the results. The cost of complex expressions is estimated recursively using a cost model over statistics about sub-expressions as well as distinct subjects and objects appearing in these results. Dynamic programming is used to enumerate different plans in order to identify the optimal one with respect to the cost model. Subtrees of a query that consists of triple patterns connected with inner joins are identified and separately optimized (i.e., find an equivalent join tree with the minimum cost). The enumeration of all possible plans is exponential, the algorithm prunes in each step the inferior plans in order to keep the enumeration space as small as possible. Semagrow operates in an asynchronous and non-blocking way where operators subscribe to a stream and are notified when data becomes available.

Lusail [53] optimizes query execution using subquery ordering based on cardinality estimation on subqueries and projection list. Cost estimation is based on statistics collected at run-time on each triple pattern during query analysis. The generated decompositions lead to a set of subqueries with minimal execution cost. Using cardinality information of individual triple patterns, Lusail estimates the cardinality of subqueries and projection list. In addition, Lusail optimizes query execution by parallelism via process scheduling. Odyssey [50] minimizes the number of subqueries that are posed to a source by combining subqueries that can be evaluated over exactly by the same sources. First, it identifies an ordering of the triple patterns within each star-shaped subquery using Characteristic Set statistics. Then, cardinality of each subquery is estimated and a dynamic programming based algorithm is applied to identify a query plan. The cost function is defined based on the cardinalities of intermediate results and on how many results need to be transferred from sources during execution. Although in an ideal scenario, Odyssey may identify efficient query plans, collecting these detailed statistics is nearly impossible in a federated scenario where datasets are autonomous. Different join methods are used by federated query processing engines. DARQ [62] and ADERIS [63] are exemplar federated query engines that makes use of NLJ (Nested-loop Join) method. Bound-join method is used by FedX, ANAPSID, SPLENDID, ADERIS, and DARQ. While FedX employ SPARQL UNION operator for sending multiple bindings to the inner sub-queries, ANAPSID, SPLENDID, ADERIS, and DARQ uses SPARQL FILTER operator. In addition to Bound-join, ANAPSID utilizes agjoin [34] method. Most optimization techniques used by the reviewed state-of-the-art federated query processing engines disregard the heterogeneity of data sources in the federation and assumes sources are homogeneous in terms of data model, access interface, privacy and access policies, and query languages. Disregarding such characteristics of data sources may lead to an execution plans that are increase execution time as well as producing incomplete answers. Furthermore, especially if the federation is composed of data sources that contain sensitive data, the generated plan may violate the privacy and access regulation set by the data providers, as a result produces incomplete results. In this thesis, we follow a heuristic-based optimization technique that utilizes the RDF-MT based source descriptions to generate an execution plan that consider the heterogeneity of data sources and generate a valid plan that respects the privacy and access policies.

3.3 Query Processing over Heterogeneous Data Sources

Existing solutions to the problem of query processing over federated data sources rely on a unified interface that allows for executing queries on a federation of homogeneous data sources in a way that execution time is minimized while query completeness is maximized. Federated database engines utilize relational model to represent the unified view of the federation and used SQL query language to express queries against the federation. Data sources in the federation have different level of heterogeneity, such as technical, data model, access method, data processing capabilities, query language, etc. Multidatabase systems have been proposed to overcome technical heterogeneity. They differ from distributed database systems in a higher degree of autonomy. Dealing with different kinds of data sources introduces new challenges to query processing. The capabilities of the DBMSs may be different, e.g. some systems may support complex queries including joins and aggregations while others do not. Traditional Data Warehouses, unlike Data Lakes, are centralized data stores that ingest data from heterogeneous data sources after transforming them in a common predefined structure. Since this kind of data integration may lead eventually to information silos, more flexible data integration approaches have been introduced in recent years. To tackle the data integration problem of heterogeneous data, a few Data Lake systems have been proposed, mainly with focus on data ingestion and metadata extraction and management. For instance, GEMMS (Generic and Extensible Metadata Management System) [64] for Data Lakes extracts metadata from heterogeneous sources, stores it in an extensible metamodel, and enriches it with semantic annotations in order to provide basic querying support. A few other approaches like SeBiDA [65] and Personal Data Lakes [66] propose to keep data from various data sources in raw format in the Data Lake after serializing them in a common data format. PolyWeb [67] and BigDAWG [68] keep data sources in raw format, i.e., without serializing them in a common data format. In PolyWeb, SPARQL queries are translated to the native query language of the sources. PolyWeb indexes each data source predicates for query decomposition and creates left-deep plans. Albeit efficient, existing approaches are not able to exploit knowledge about the main features of the integrated data sources, and produce query plans *customized* for sources selected for collecting the data from the Data Lake.

3.4 Privacy-aware Federated Query Processing

The data privacy control problem has received extensive attention by the database community; approaches by De Capitani et al. [69] and Bater et al. [70] are exemplars that rely on an authority network to produce valid plans. Albeit relevant, these approaches are not defined for federated query processing systems; thus, the tasks of source selection and query decomposition are not addressed. The Semantic Web community has also explored access control models for SPARQL query engines; RDF named graphs [71–73] and quad patterns [74] are used to enforce access control policies. Most of the work focuses on the specification of access control ontologies and enforcement on RDF data [71, 73] stored in a centralized RDF store, while others explore access control specification and enforcement on distributed RDF stores [75, 76] and federated query processing [72, 77] scenarios. Costabello et al. [71] present SHI3LD, an access control framework for RDF stores accessed on mobile devices; it provides a pluggable filter for generic SPARQL endpoints that enforces context-aware access control at named graph level. Kirrane et al. [74] propose an authorization framework that relies on stratified Datalog rules to enforce access control policies; RDF quad patterns are used to model permissions (grant or deny) on named graphs, triples, classes, and properties. Ubehauen et al. [73] propose an access control approach at the level of named graphs; it binds access control expressions to the context of RDF triples and uses a query rewriting method on an ontology for enabling the evaluation of privacy regulations in a single query. SAFE [72] is designed to query statistical RDF data cubes in distributed settings and also enables graph level access control. In this thesis, we proposed a privacy-aware federated engine where policies are defined over a privacy-aware RDF-MTs (PRDF-MTs). It enables access control statements to be defined at different level of granuality, i.e., at the source level as well as at the mediator level. Our approach generates execution plans that both enforce privacy regulations set by the data sources and speed up execution time.

3.5 Update Propagation and Co-evolution

Many data products and services rely on full or partial local replications of data sources to ensure faster query processing. Most related work on dataset update propagation focuses on distributed publish/subscribe systems [78, 79], resource link maintenance [80, 81], target synchronization [82], partial replicas [83], data-shipping [84], lazy updates [85], and real-time update notification [81, 86]. In [78], the authors propose a peer-to-peer publish/subscribe system for events described in RDF. By avoiding the use of multiple indexes for the same publication, they manage to reduce storage space. Similarly, [79] provide an implementation with publish/subscribe capabilities in an RDF-based peer-to-peer system to manage digital resources. As for resource link maintenance, *DSNotify* [80] offers a change-detection framework to detect and fix broken links between resources in two datasets while, *Semantic Pingback* [81] proposes a notification system for the creation of new links between Web resources. To note that this approach is suitable for relatively static resources, i.e., RDF documents or RDFa annotated Web pages. In contrast, *SparqlPuSH* [86] offers a real-time notification framework for data updates in a RDF store using a semantic *PubSubHubbub*-based protocol (PuSH). SparqlPuSH allows users to subscribe for changes updates of a subset of content in a RDF store using SPARQL. However, notification and broadcasting are only available as RSS and Atom feeds. As regards target synchronization, *RDFSync* [82] performs update synchronization by merging source and target graphs to get the updated target RDF graph. Alternatively, [83] has designed an approach to replicate, modify, and write-back parts of an RDF graph on devices with low computing power. However, this approach does not resolve conflicts arising with concurrent modifications on both the base graph and the partial replicas. Bleiholder et.al., [87] classifies conflict resolution strategies and presents a catalog of resolution functions. They divide the conflict resolution strategies into three classes: ignorance, avoidance, and resolution. Conflict ignorance strategies are not aware of conflicts in the data. Conflict avoidance strategies are aware of whether and how to handle inconsistent data. Conflict resolution strategies may use metadata to resolve conflicts. These can be divided into deciding and mediating; deciding strategy chooses value from already existing values whereas a mediating strategy may compute a new value. In distributed databases, where data is replicated on different sites, Lazy update protocols [85] disseminate updates to replicas to ensure consistency. These protocols guarantee serializable execution as well as high performance.

CHAPTER 4

Data Source Description Model

This chapter addresses the challenge of describing heterogeneous data sources in a Semantic Data Lake by capturing the semantics encoded in them. Different data sources provide different data structuredness, querying capabilities, access interfaces, and restrictions. Federated query processing systems need to consider these differences while executing queries. Describing such heterogeneity helps to understand data source capabilities and to help different applications, especially federated query processing systems, discover relevant sources for processing and interpret them accordingly. The content of this chapter is based on the publications [88–91]. Figure 4.1 shows the challenge we tackled in this chapter: i.e., describe knowledge represented in a federation of heterogeneous data sources in a Semantic Data Lake. The result of this chapter provide an answer to the following research question:

> RQ1: How can we describe the semantics encoded in heterogeneous data sources?

To represent knowledge that exist in heterogeneous data sources, a data source description model that describe the semantics, i.e., the meaning of data points, represented in those sources is crucial. The Semantic Web community provide several standards and specifications to represent a piece of data, such as RDF data model and ontology definition language such as RDFS and OWL. Our data source description model is based on RDF data model. Our assumption is that data sources are either inherently represented in RDF data model or the mapping from non-RDF data model to RDF is available. Creating mappings for non-RDF data is out of scope of this thesis.

We summarize the contributions of this chapter as follows:

- A novel data source description model that exploits the semantics encoded in heterogeneous data sources.
- A novel algorithm to create RDF-Molecule Template based description of heterogeneous data sources.
- A detailed analysis of different benchmarks of federation of data sources using RDF-Molecule Template based descriptions.
- An empirical evaluation of impact of data source descriptions computed using different graph partition methods for query processing.

This chapter is structured as follows: First, in Section 4.1, we present a motivating example illustrating the problem of data source description and the effect of, or lack there of, expressiveity in federated

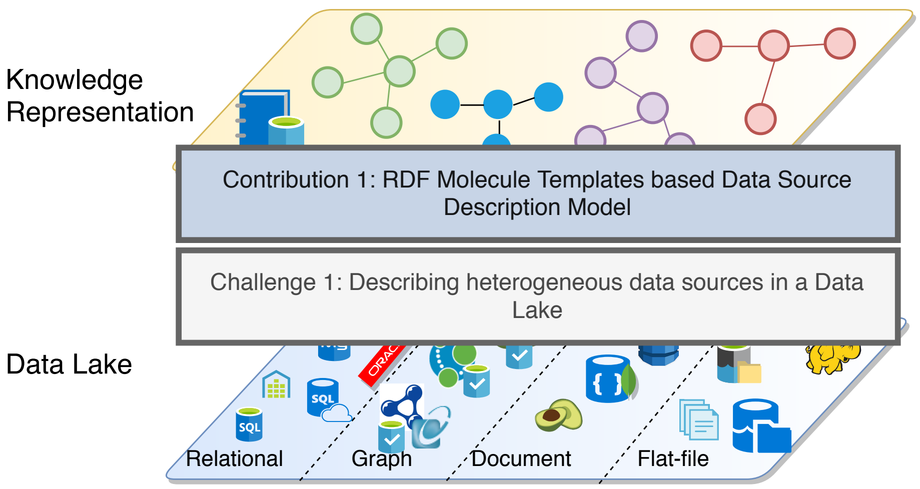

Knowledge Representation
Contribution 1: RDF Molecule Templates based Data Source Description Model
Challenge 1: Describing heterogeneous data sources in a Data Lake
Data Lake
Relational
Graph
Document
Flat-file

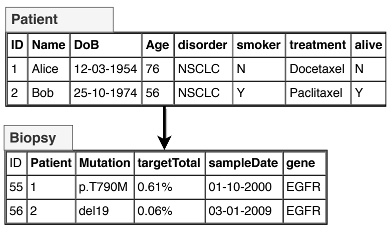
Figure 4.1: **Challenges and Contributions**.This chapter focuses on the problem of describing heterogeneous data sources for data integration, and propose RDF Molecule Templates based data source description for federated query processing

query processing. Next, in Section 4.2, we describe our proposed data source description model. To address research question **RQ1**, we propose RDF Molecule Templates, an abstract representation of entities that belong to same semantic concepts and their relationships, that represent the knowledge in the heterogeneous data sources to be exploited by different sub-problems of a federated query processing system. An analysis of three benchmarks for federation of data sources using RDF-MT based data source description and comparison of data source descriptions computed using different graph partitioning methods is presented in Section 4.3. Observed results suggest that RDF-MT based data source description are able to unveil patterns that represent the semantic knowledge represented in each data sources of a federation as well as the connectivity between concepts in each data source and with other data sources in the federation. In addition, such knowledge representation model present an opportunity to provide services such as exploration and extraction of fragments of knowledge. Furthermore, the RDF-MT based source descriptions improved the performance of federated query processing system compared to the data sources computed using graph partitioning methods. Finally, Section 4.4 presents the closing remarks of this chapter.

4.1 Motivating Example

The discovery of relevant datasets for different application such as federated query processing, question answering, and entity linking has becoming challenging, due to growth in the amount of available big data sources. Such applications must be aware of the available data sources, the content of each data source, their capabilities, and access mechanism of the sources in order to select relevant sources, process query over them, and interpret the results. For federated query processing, data source description is used for selecting relevant data sources that can contribute to answering a query posed by the user. Such description may also contain a mapping between source schema to some ontology terms, if the data sources do not naively support RDF data model. These mappings guide the system to automatically select and semantify the available data on-the-fly.

Consider, for example, the data sources presented in Figure 4.2. The first data source, S_1 in Figure 4.2a, shows information stored in a hospital database about patients demography, diagnosis, treatment and prognosis represented in relational model. Data is stored in a relation data management system (RDBMS),



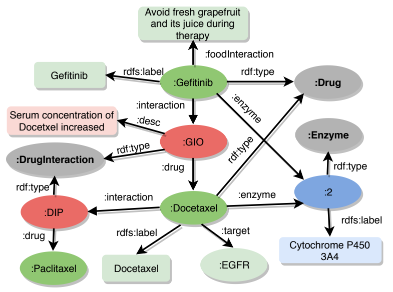
Patient

ID	Name	DoB	Age	disorder	smoker	treatment	alive
1	Alice	12-03-1954	76	NSCLC	N	Docetaxel	N
2	Bob	25-10-1974	56	NSCLC	Y	Paclitaxel	Y

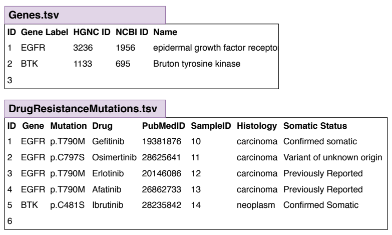
Biopsy

ID	Patient	Mutation	targetTotal	sampleDate	gene
55	1	p.T790M	0.61%	01-10-2000	EGFR
56	2	del19	0.06%	03-01-2009	EGFR

(a) S_1: Relational Tables



(b) S_2: RDF Data



Genes.tsv

ID	Gene Label	HGNC ID	NCBI ID	Name
1	EGFR	3236	1956	epidermal growth factor recepto
2	BTK	1133	695	Bruton tyrosine kinase
3				

DrugResistanceMutations.tsv

ID	Gene	Mutation	Drug	PubMedID	SampleID	Histology	Somatic Status
1	EGFR	p.T790M	Gefitinib	19381876	10	carcinoma	Confirmed somatic
2	EGFR	p.C797S	Osimertinib	28625641	11	carcinoma	Variant of unknown origin
3	EGFR	p.T790M	Erlotinib	20146086	12	carcinoma	Previously Reported
4	EGFR	p.T790M	Afatinib	26862733	13	carcinoma	Previously Reported
5	BTK	p.C481S	Ibrutinib	28235842	14	neoplasm	Confirmed Somatic
6							

(c) S_3: Flat Files

Figure 4.2: **Motivating Example.** Heterogeneous data sources that are connected to each other via drug information. a) data source that contain information about patients demography, diagnosis, treatment, and prognosis stored in relational database. b) a data source that contains information about drug chemical properties, interactions, enzymes and targets stored in RDF data store, aka Triple Store. c) data source contains information about genomic analysis, specifically gene names, mutations, information about whether these mutations are resistant to specific drugs, and reference to scientific publications from PubMed stored as a tab-separated flat file. The challenge tackled in this chapter focus on describing these data sources to represent the knowledge encoded in them semantically to facilitate an efficient data integration and exploration.

e.g., MySQL, that is optimized to handle this type of data model. To access this data, one should write SQL queries over the given relations. Furthermore, since the data is highly sensitive, only authorized users can access such data especially patient's Name, SSN, and date of birth. The second data source, Figure 4.2b, provide data in RDF (graph) data model about drugs, enzymes, and drug-drug interactions. Data is stored in a triplestore, e.g., Virtuoso, that is optimized to handle data in triples (subject, predicate, object) form. Triplestores provide a SPARQL endpoint service that allow users to access the data by writing SPARQL queries. Finally, the third data source, Figure 4.2c, shows genomic analysis data stored in a tab-separated flat file about tumor samples, mutations, genes, and whether mutation confers drug resistance reported in scientific publications in PubMed[1]. Since the data is not stored in a database management system, there is no querying mechanism available inherently. Thus, special software component is needed to parse and extract data from it.

Though the data sources in Figure 4.2 have different data representation model, they contain inter-related information. For instance, Alice was diagnosed with non-small cell lung cancer (`NSCLC`) where biopsy results show that *EGFR* mutated and *p.T790M* mutation was found; treated with *Docetaxel* chemotherapy drug. One can verify that the mutation type found, i.e., p.T790M, is not resistant to the treatment drug, i.e., Docetaxel. On the other hand, this tumor type was reported to be resistant to *Gefitinib* drug, in S_3 (Figure 4.2c). In addition, these two drugs should not be administered at the same time since they interact with each other, i.e., the serum concentration of *Docetaxel* can be increased when it is combined with *Gefitinib*, in S_2 (Figure 4.2b). Performing such analysis manually is not feasible in the context of big data, automated systems are needed to discover and integrate data. To access these data sources in a federated fashion, they should first be described in a way that facilitate the integration of disparate data to a common knowledge representation model. Semantic Web technologies can be utilized to describe the knowledge represented in disparate data sets.

[1] `https://www.ncbi.nlm.nih.gov/pubmed/`

4.2 Source Description Model

Source descriptions specify information about available data points, capabilities, and access methods. Available data points can be described using a shared semantic concept and set of predicates associated to them. Our data source description model is composed of three basic components: schema mapping, privacy and access control policies, and access methods of data sources. *Schema mapping* describes the mapping between the source schema to global schema [3]. We follow the global-as-view (GAV) approach and propose RDF Molecule Templates (RDF-MTs) that are considered as the global schema of the federation, and each data source schema is mapped with set of RDF Molecule Templates, as described in Section 4.2.1. *Privacy and access control policies* define the restriction on data points and who can and cannot access as well as what kind of operations can be performed on them. Such policies are embedded in RDF Molecule Templates, as defined in Definition 27. Details on how privacy and access control policy description are used by privacy-aware federated query processing techniques is presented in Chapter 7. Finally, the third component, *access methods of data sources*, provides information about the access URL, query language, and method to be used while contacting the sources via the given URL.

4.2.1 RDF-MT: RDF Molecule Templates

Our source description model is based on the concept of RDF molecules; a set of triples that share same subject values are called *RDF Molecules*. Formally, RDF molecules are defined as follows:

Definition 10 (RDF Molecule [23]) *Given an RDF graph G, an RDF molecule $\mu \subseteq G$ is a set of triples $\mu = \{t_1, t_2, \ldots, t_n\}$ in which $subject(t_1) = subject(t_2) = \cdots = subject(t_n)$.*

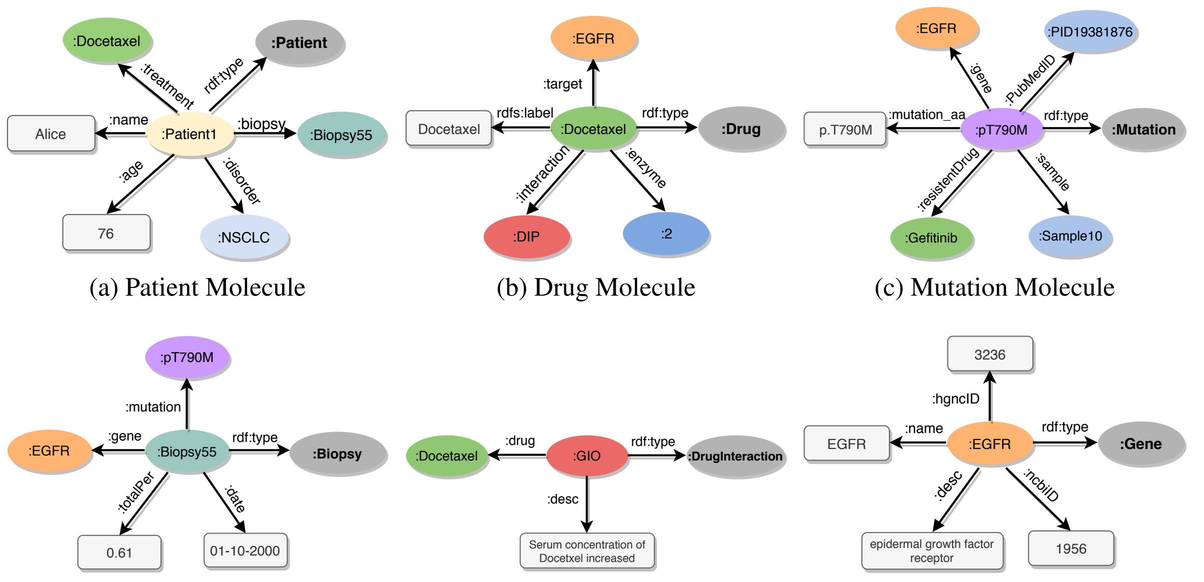
Given a set of heterogeneous data sources, $\mathbb{F} = \{S_1, S_2, S_3\}$, and a set of mapping files, $M = \{M_{S_1}, M_{S_3}\}$, for each non-RDF data source, Figure 4.2 presents example of RDF molecules: `Patient` and `Biopsy` RDF molecules from relational data source S_1, `Drug` and `DrugInteraction` RDF molecules from RDF data source S_2, and finally `Mutation` and `Gene` RDF molecules extracted from flat (TSV) files S_3 are shown in Figure 4.2.

In this thesis, we tackle the problem of source descriptions for federated query processing using the concept of RDF molecule templates that encode an abstract description of a set of data points that share similar characteristics such as semantic type of entities. An RDF molecule template (RDF-MT) is formally defined as follows:

Definition 11 (RDF Molecule Template (RDF-MT)) *An RDF Molecule Template (RDF-MT) is a 5-tuple=<M,C,DTP,IntraL,InterL>, where:*

- *M – is a set of mappings from source schema to molecule predicates, DTP;*
- *C – is an RDF class such that the triple pattern (?s T_p C) is true in G, where T_p is a typing predicate such as $rdf : type$, or $wdt : P31$;*
- *DTP – is a set of pairs (p, T) such that p is a property with domain C and range T, and the triple patterns (?s p ?o), and (?s T_p C) are true in G;*
- *IntraL – is a set of pairs (p,C_j) such that p is an object property with domain C and range C_j, and the triple patterns (?s p ?o) and (?o T_p C_j) and (?s T_p C) are true in G;*
- *InterL – is a set of triples (p,C_k,SW) such that p is an object property with domain C and range C_k; SW is a URL that provides access to an dataset K, the triple patterns (?s p ?o) and (?s T_p C) are true in G, and the triple pattern (?o T_p C_k) is true in K.*

Privacy-aware RDF Molecule Templates can be employed for describing and enforcing privacy policies. Privacy and access control policies can be encoded in RDF-MTs at level of the predicates or RDF-MTs



(a) Patient Molecule (b) Drug Molecule (c) Mutation Molecule

(d) Biopsy Molecule (e) Drug Interaction Molecule (f) Gene Molecule

Figure 4.3: **Example RDF Molecules.** RDF molecules represent a set of triples that share the same subject, hence represented as star-shaped RDF graphs. a) Presents `:Patient1` RDF molecule that represent (semantically) the first row of `Patient` table in Figure 4.2a. b) `:Docetaxel` RDF molecule in Figure 4.2b. c) `:pT790M` RDF molecule that represent (in RDF) the first row of *DrugResistnceMutation.tsv* file in Figure 4.2c.

Algorithm 1 Create RDF Molecule Templates: WI: Set of pairs (ws, M), and WAI: hash map of $ws \in WI$ to RDF Molecule Templates

```
 1: procedure CreateMoleculeTemplates($WI$)
 2:     $WAI \leftarrow \{\}$                                        ▷ $WAI$ - a map of $ws$ and $MTLs$
 3:     for $ws_i, M \in WI$ do
 4:         $WAI(ws_i) \leftarrow CollectRDFMoleculeTemplates(ws_i, M)$          ▷ Algorithm 2
 5:     end for
 6:     for $(ws_i, [(C_i, \mathbb{P}_i, \mathbb{L}_i)]) \in WAI$ do
 7:         for $(ws_k, [(C_k, \mathbb{P}_k, \mathbb{L}_k)]) \in WAI \textit{ and } ws_k \neq ws_i$ do
 8:             $\mathbb{L}_i \leftarrow \mathbb{L}_i + (p_i, C_k) \textit{ such that } \exists p_i \in \mathbb{P}_i \wedge range(p_i) = C_k$
 9:         end for
10:     end for
11:     saveRDFMT($WAI$)
12: end procedure
```

by using different access control theory operators. Details on available access control theory operators and privacy-aware RDF Molecule Templates is presented in Chapter 7.

4.2.2 Creating RDF Molecule Templates

Given a set of data sources in a federation, Algorithms 1 and 2 create RDF Molecule Template based source description. Given a set of pairs (ws, M) where ws is access interface of a data source and M is a set of mapping rules, Algorithm 2 extracts RDF-MTs by either executing queries against a SPARQL endpoints of RDF datasets or parsing the given mapping rules of a non-RDF data source. First, *RDF*

Algorithm 2 Collect RDF Molecule Templates: ws: data access URL, and M: mapping rules, if available

```
 1: procedure CollectRDFMoleculeTemplates($ws$, $M$)
 2:     $MTL \leftarrow [\,]$                                    ▷ $MTL$ - list of molecule templates
 3:     $CP \leftarrow getClassesWithProperties(ws)$
 4:     for $(C_i, \mathbb{P}_i) \in CP$ do                      ▷ $C_i$ - class axioms
 5:         $\mathbb{L}_i \leftarrow \mathbb{L}_i + (p_i, C_j) | \exists p_i \in \mathbb{P}_i, (range(p_i) = C_j) \wedge ((C_j, \mathbb{P}_j) \in CP)$
 6:         $RDF\text{-}MT_i \leftarrow (C_i, \mathbb{P}_i, \mathbb{L}_i)$
 7:         $MTL \leftarrow MTL + RDF\text{-}MT_i$
 8:     end for
 9:     return $MTL$
10: end procedure
```

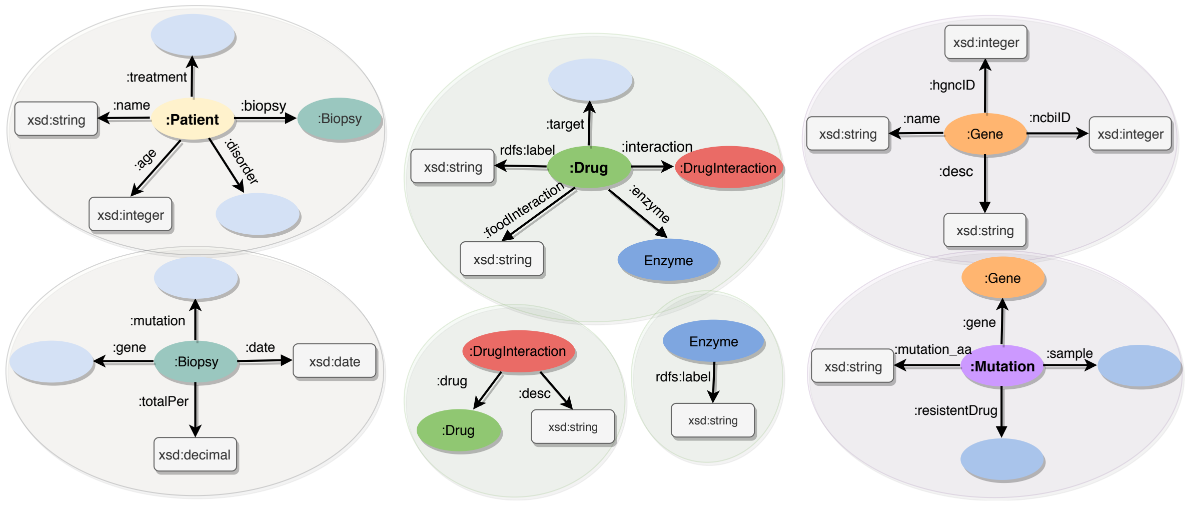
classes and their corresponding *properties* are collected (Line 3), i.e., pairs $(C_i, \mathbb{P}_i)$, where C_i is a class and $\mathbb{P}_i$ is a set of predicates of C_i. RDF class typing predicate is defined in access method description as input to the algorithm. Then, for each RDF class C_i and their corresponding object properties in $\mathbb{P}_i$ (i.e., properties that have URI as values), intra-dataset links (IntraL) are generated (Line 4–8). Intra-links represent links between RDF-MTs within the same data source.

Algorithm 1 first collect a list of RDF-MTs and their intra-dataset links within each data sources by calling Algorithm 2 (Line 3–5). Then it iterates over each RDF-MT in each Web access interface (WAI) and finds inter-dataset links between them (InterL) (Line 6–10). Inter-dataset links represented by links between RDF-MTs in two or more data sources. Finally, RDF-MTs are stored to as a file (Line 11).

Given data sources S_1, S_2, and S_3 in Figure 4.2 and RML mappings M_{S_1} and M_{S_3} for non-RDF data sources, Figure 4.4 and 4.5 illustrates the RDF molecule template creation process using the algorithms described above. Figure 4.4 shows six RDF molecule templates and their associated predicates identified using Algorithm 2. We set the typing predicate in all three sources as `rdf:type`. `:Patient` and `:Biopsy` RDF-MTs are created from the first source and contains a maximum of five, respectively four, properties associated with them. Since this data source is a non-RDF source, mapping rules (M_{S_1}) are used to represent the semantic meaning of tables and column names. A snippet of these mapping rules is shown in Listing 4.1. Using this mapping rules, the algorithm contact the data sources to find the links if not already represented by the rules. For instance, the range of *:treatment* and *:disorder* properties are not defined by the rule, hence the algorithm has to execute query to find internal as well as external links with other RDF molecule templates [2]. Notice that at this stage of the algorithm there is only one (intra-) link identified in this data source, i.e., `:Patient` to `:Biopsy`. Unknown links are illustrated with nodes without name, e.g., object type of `:treatment` and `:disorder` predicates of the `:Patient` RDF molecule template are not known.

Three RDF molecule templates, `:Drug`, `:Enzyme` and `:DrugInteraction`, are created from the second source, S_2, each contain five, two and one predicates, respectively. Unlike S_1 which store relation tables, S_2 provide RDF data and the algorithm executes SPARQL queries to extract RDF molecule template predicates and intra-links within the data source. Consequently, three intra-dataset links are identified, i.e., bidirectional link between `:Drug` and `:DrugInteraction` via `:interaction` and `:drug`, respectively, and from `:Drug` to `:Enzyme`. From the third data source, S_3, `:Gene` and `:Mutation` RDF molecule templates are created, each contains a maximum of four predicates. There is only one intra-dataset link identified in this source, i.e., from `:Mutation` to `:Gene`. Note that, information about the `Sample` where the mutation is studied which contains the `Histology` and `Somatic Status` is not represented by the given mapping rules, hence no RDF molecule template is

[2] For consistency reasons of the illustrations, we omitted prefix `onto`



(a) S_1: RDF Molecule Templates (b) S_2: RDF Molecule Templates (c) S_3: RDF Molecule Templates

Figure 4.4: **RDF Molecules Templates per Data Source.** a) Shows RDF-MTs created from S_1 and intra-link from `:Patient` RDF-MT to `:Biopsy` RDF-MT via `:biopsy` property. b) Shows RDF-MTs created from S_2 and intra-links from `:Drug` RDF-MT to `:DrugInteraction` RDF-MT via `:interaction` property, from `:DrugInteraction` RDF-MT to `:Drug` RDF-MT via `:drug` property, and from `:Drug` RDF-MT to `:Enzyme` RDF-MT via `:enzyme` property. c) Presents RDF-MTs created from S_3 and intra-link from `:Mutation` RDF-MT to `:Gene` RDF-MT via `:gene` property.

created for it.

Listing 4.1: M_{S_1} RML Mapping

```
:DB_source   d2rq:jdbcDSN      "patientDB";
       d2rq:jdbcDriver  "com.mysql.cj.jdbc.Driver";
       d2rq:username     "root";   d2rq:password  "" .
<#TM1>   rml:logicalSource [
     rml:source  :DB_source; rr:sqlVersion  rr:SQL2008; rr:tableName "Patient
        "];
   rr:subjectMap  [
     rr:template  "http://hsp1.org/Patient/{ID}";
     rr:class  onto:Patient     ];
   rr:predicateObjectMap  [
     rr:predicate  onto:name;
     rr:objectMap  [rml:reference  "Name"]];
   rr:predicateObjectMap  [
     rr:predicate  onto:age;
     rr:objectMap  [rml:reference  "Age"]];
   rr:predicateObjectMap  [
     rr:predicate  onto:treatment;
     rr:objectMap  [rml:template  "http://example.org/drugs/{treatment}"]];
   rr:predicateObjectMap  [
     rr:predicate  onto:disorder;
     rr:objectMap  [rml:template  "http://example.org/disease/{disorder}"
        ]];
   rr:predicateObjectMap  [
     rr:predicate  onto:biopsy;
     rr:objectMap  [  rr:parentTriplesMap  <#TM2>  .
```

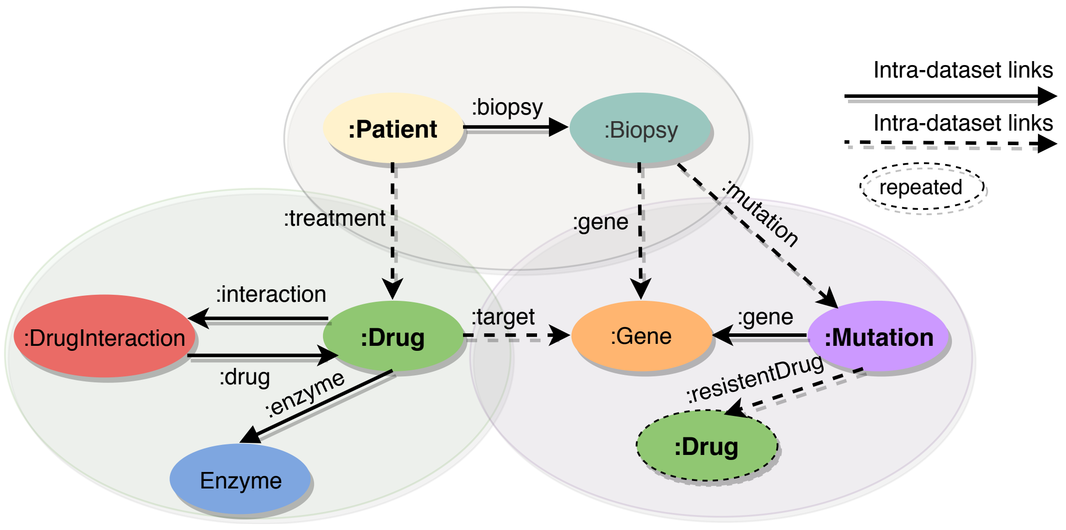


Figure 4.5: **Intra- and Inter-dataset Links:** Each data source is represented by oval shapes that contains each RDF-MTs and Intra-dataset links between them within the same data source (represented by solid line). Inter-dataset links represent links between RDF-MTs that identified in different data sources (represented in dashed links).

```
        rr:joinCondition [ rr:child "ID";  rr:parent "Patient"
      ]  ]  ].
  <#TM2>  rml:logicalSource [
      rml:source  :DB_source; rr:sqlVersion  rr:SQL2008; rr:tableName  "Biopsy"
         ];
    rr:subjectMap [
      rr:template "http://hsp1.org/Biopsy/{ID}";
      rr:class  onto:Biopsy ];
    rr:predicateObjectMap [
      rr:predicate  onto:mutation;
      rr:objectMap [rml:template "http://example.org/mutation/{Mutation}"
         ]];
      .....
```

Figure 4.5 shows inter-links between RDF-MTs of different data sources. As can be seen, there is an external link from `:Patient` molecules in S_1 data source to `:Drug` molecules in S_2 data source via `:treatment` predicate. Additionally, there are two external links from `:Biopsy` molecules in S_1 data source to `:Gene` molecules via `:gene` predicate and to `:Mutation` molecules via `:mutation` predicate in S_3 data source. There is a bidirectional link between S_2 and S_3 data sources. `:Drug` molecules in S_2 are connected to `:Gene` molecules in S_3 via `:target` predicate and `:Mutation` molecules in S_3 are connected to `:Drug` molecules in S_2 via `:resistentDrug` predicate. These links represent the existence of joins between a set of RDF molecules in one dataset to another dataset in the federation. In our running example, there are three data sources that are members of the federation. We can observe that the description of data sources not only describe the schema of each data sources and their mapping to the global schema, RDF-MTs in this case, but also the existence of connections between data sources at the level of RDF molecule templates.

4.3 Experimental Study

We analyze three different benchmarks for federated query processing to show case the expressiveness of RDF Molecule Template based data source descriptions. Furthermore, we assess the performance of a federated query engine utilizing RDF Molecule Templates and templates generated using different graph partitioning algorithms. The following research questions are evaluated: **Q1)** Do RDF-MTs characterize the semantics represented within and between data sources? **Q2)** Do different source descriptions impact on federated query processing in terms of efficiency and effectiveness?

4.3.1 RDF-MT based Characterization of Benchmarks

Three benchmarks are utilized to assess our research questions: i) BSBM - The Berlin SPARQL Benchmark; ii) FedBench; and iii) LSLOD. For each RDF dataset in the benchmarks, the RDF-MTs are computed; furthermore, graph analytics are utilized to describe the properties of these datasets in terms of the RDF-MTs and the connection between them. We generated all RDF-MTs and their interconnections considering both intra-dataset and inter-dataset links, as defined in Algorithm 2 and Algorithm 1, respectively. We use graph density, connected components, transitivity, and average clustering coefficient to analyze the main properties of the graph that model the RDF-MTs of each federation. Clustering coefficient measures the tendency of nodes who share same connections in a network to become connected. If the neighborhood is fully connected, the clustering coefficient is 1 and a value close to 0 means that there are no connections in the neighborhood. Average clustering coefficient assigns higher scores to low degree nodes, while the transitivity ratio places more weight on the high degree nodes. The average connectivity of a graph is the average of local node connectivity over all pairs of nodes of the graph. We model a graph of RDF-MTs as undirected a multi-graph (MultiGraph in networkx[3]) where the predicates that connect each RDF-MTs are used as labels of the edges. A multi-graph is used to compute the number of nodes, edges, average number of neighbors, connected components, and average node connectivity. Finally, we model the graph as undirected single network graph, where a link between RDF-MTs is represented as unlabelled edges. Using single network graphs, transitivity and average clustering coefficient are computed.

BSBM - The Berlin SPARQL Benchmark

The Berlin SPARQL Benchmark [92] is a synthetic dataset focusing on an e-commerce use case where a set of products are offered by different vendors and consumers and reviewers have posted reviews about these products on various review sites. The data model contains eight classes: Product, ProductType, ProductFeature, Producer, Vendor, Offer, Review, and Person.

The first benchmark for our experiment, and the smallest in number of RDF-MTs, is the Berlin SPARQL Benchmark (BSBM). In BSBM benchmark, there are only eight RDF-MTs and eight links between them. For our experiment we treated each RDF-MT as a single dataset by creating a separate endpoint for them. Therefore, there are in total eight datasets and since each dataset contains only one RDF-MT, there are no intra-dataset links for this benchmark. Figure 4.6 illustrates all RDF-MTs in BSBM where each contained in a single dataset (hence different colors) and their inter-dataset connections[4]. In addition, in order to study the characteristics of the generated BSBM molecule template graph, we report on a graph analysis in Table 4.1. We observed a strong connection between RDF-MTs - 0.285 density

[3] `https://networkx.github.io/`

[4] The graph visualization was generated using the open source software platform cytoscape – `http://www.cytoscape.org/`

Num of nodes	8
Num of edges	8
Graph density	0.285
Avg. num of neighbors	2
Connected components	1
Avg. node connectivity	1.0
Transitivity	0.0
Clustering coefficient	0.0

Table 4.1: **FedBench RDF-MT Graph Metrics**. Clustering coefficient (0.0) suggests that there is no connectivity in the neighborhood of the network.

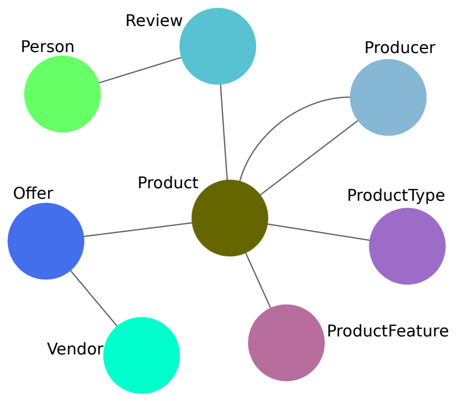


Figure 4.6: **Analysis of RDF-MTs of BSBM**. The graph comprises 8 RDF-MTs and 8 inter-dataset links. Each dot represents an RDF-MT stored in each endpoint. A line between dots corresponds to inter-dataset links. There is only one RDF-MT in each endpoint, hence no intra-dataset links.

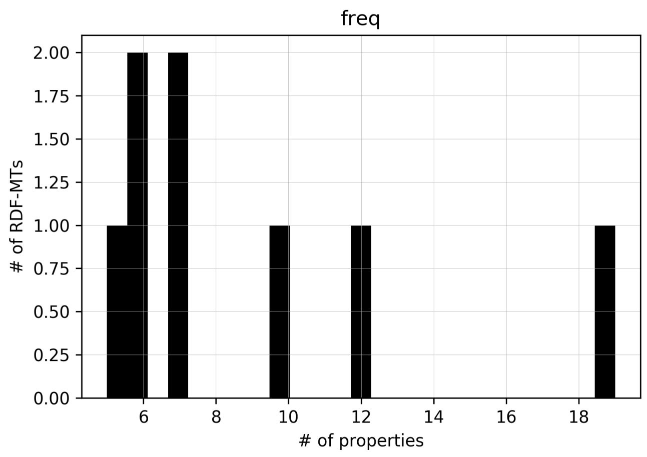


Figure 4.7: **Frequency of BSBM RDF-MTs Per Number of Properties**. Majority of Molecule Templates contain from five to seven properties.

and 1.0 average node connectivity. In particular, the connections concentrated on a single RDF-MT (hence, a single dataset), `Product`, with 6 out of 8 links to or from this RDF-MT (hence, dataset). A histogram of frequencies of RDF-MTs per numeber of properties distributed from six (two RDF-MTs) to 18 (one RDF-MT) is shown in Fig. 4.7.

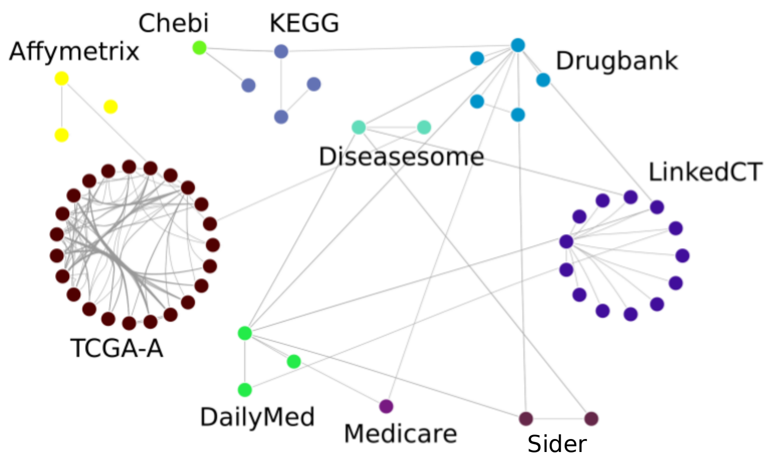


Figure 4.8: **Analysis of RDF-MTs of LSLOD**. The graph comprises 56 RDF-MTs and 197 intra- and inter-dataset links; dots in each circle represent RDF-MTs. A line between dots in the same circle shows intra-dataset links, while a line between dots in different circles corresponds to an inter-dataset link. There are nine datasets: Drugbank, Dailymed, Sider, Affymetrix, KEGG, LinkedCT, TCGA-A, ChEBI, and Medicare; they have six, three, two, three, four, 13, 23, one, and one RDF-MTs, respectively.

Num of nodes	57
Num of edges	197
Graph density	0.205
Avg. num of neighbors	11.474
Connected components	3
Avg. node connectivity	1.648
Transitivity	0.634
Clustering coefficient	0.375

Table 4.2: **LSLOD RDF-MT Graph Metrics**. Clustering coefficient (0.375) suggests high number of intra- & inter-dataset links.

LSLOD: Life Science Linked Open Data

LSLOD [93] is a benchmark composed of 10 real-world datasets of the Linked Open Data (LOD) cloud from life sciences domain. The federation includes: ChEBI (the Chemical Entities of Biological Interest), KEGG (Kyoto Encyclopedia of Genes and Genomes), DrugBank, TCGA-A (subset of The Cancer Genome Atlas), LinkedCT (Linked Clinical Trials), Sider (Side Effects Resource), Affymetrix, Diseasome, DailyMed, and Medicare. Compared to FedBench, LSLOD datasets contain rather small number of RDF-MTs. Figure 4.8 shows the connectivity of all RDF-MTs associated with LSLOD datasets. In total, there are 57 RDF-MTs with 197 links between them. TCGA-A dataset contains the majority of RDF-MTs (23). There are no shared RDF-MTs between the LSLOD datasets. Figure 4.9 shows that most of the RDF-MTs have between three and 55 properties. Some RDF-MTs from TCGA-A have a large number of properties, e.g., `tcga:clinical_omf` has 197 properties; `tcga:normal_control`, `tcga:tumor_sample`, and `tcga:clinical_nte` have 246 properties; and `tcga:clinical_cqcf`, `tcga:biospecimen_cqcf`, and `tcga:patient` have 247 properties. Graph analysis in Table 4.2 shows that there is medium connectivity (stronger than FedBench) of RDF-MTs, with 0.123 density, 6.912 average number of neighbors, and 3 connected components.

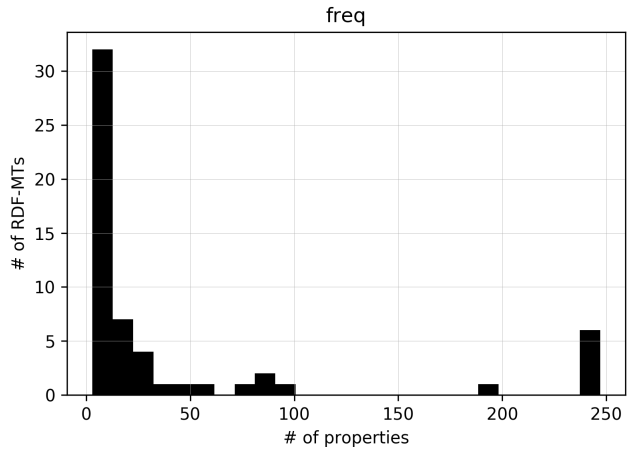


Figure 4.9: **Frequency of LSLOD RDF-MTs Per Number of Properties**. Majority of Molecule Templates contain from three to 30 properties.

FedBench

FedBench [94] is a benchmark suite for analyzing both the efficiency and effectiveness of federated query processing techniques for different use cases on semantic data. It includes three collections of datasets: **cross-domain**, **life-science**, and **SP^{2}Bench** collections. The **cross-domain** collection is composed of datasets from different domains: DBpedia has linked structured data extracted from Wikipedia; Geonames is composed of geo-spacial entities such as countries and cities; Jamendo includes music data such as artists, records; LinkedMDB maintains linked structured data about movies, actors; the New York Times dataset contains about 10,000 subject headings about people, organizations, and locations; finally, the Semantic Web Dog Food (SWDF) dataset includes data about Semantic Web conferences, papers, and authors. Furthermore, **Life-science** collection contains datasets from the life-sciences domain: Kyoto Encyclopedia of Genes and Genomes (KEGG) has chemical compounds and reactions data in Drug, Enzyme Reaction and Compound modules; the Chemical Entities of Biological Interest (ChEBI) contains information about molecular entities on “small” chemical compounds, such as atoms, molecules, ions; and DrugBank maintains drug data with drug target information. In addition to these three datasets in the life-sciences collection, a subset of DBpedia dataset that includes data about drugs is added in this collection. Finally, **SP^{2}Bench** collection contains a synthetic dataset generated by the SP^{2}Bench data generator [95], that mirrors characteristics observed in the DBLP database. For our experiments, we have used datasets from the first two collections from this benchmark, i.e., **cross-domain** and **life-science** collections, which contain real-world datasets.

In FedBench, RDF-MTs that have more than 100 properties correspond to classes with multiple predicates and subclasses, such as `dbo:Person`, `dbo:Organisation`, and `dbo:Place`. In addition, in order to study the characteristics of the generated FedBench RDF-MT graph, we report on a graph analysis which is documented in Table 4.3. In particular, we observe a rather medium connectivity of the graph nodes (i.e., RDF-MTs) – 0.081 – with 31.9 average number of neighbors and 9 connected components[5]. Finally, the clustering coefficient (0.602) indicates that we do not have only links between the RDF-MTs that come from the same dataset, but also many inter-dataset connections. Figure 4.10 illustrates all RDF-MTs in FedBench associated with the dataset they come from with all intra-dataset and inter-dataset connections. In total, 387 RDF-MTs (396 including shared RDF-MTs) with 6, 317 links

[5] A lower number of connected components indicates a stronger connectivity.

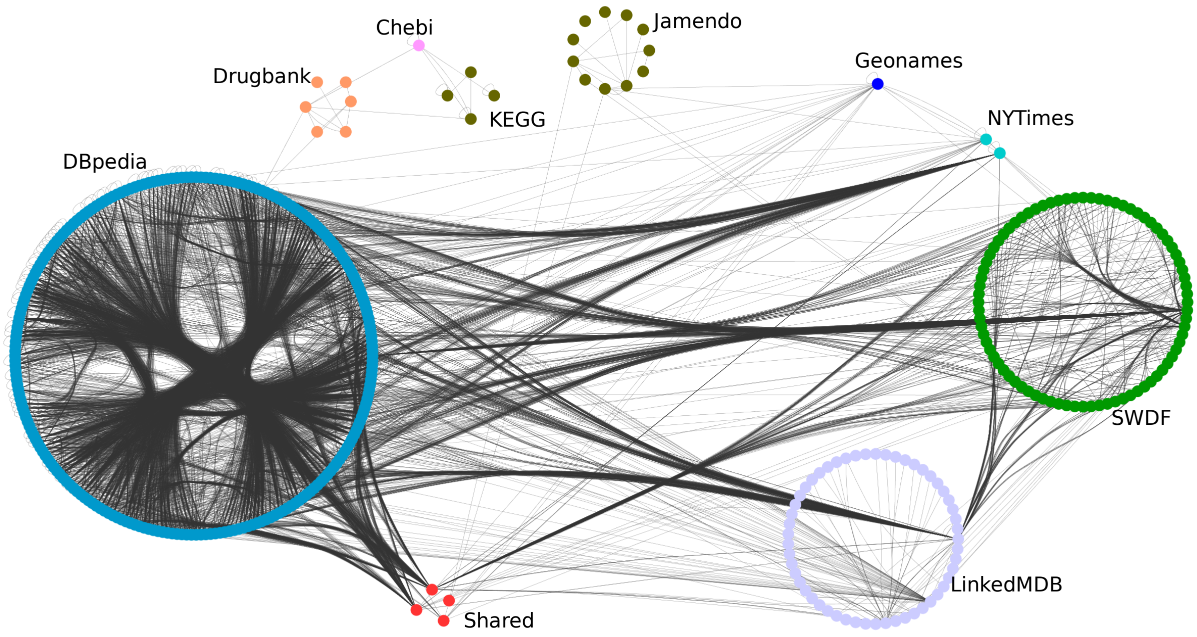


Figure 4.10: **Analysis of RDF-MTs of FedBench**. The graph comprises 387 RDF-MTs and 6,317 intra- and inter-dataset links. The dots in each circle represent RDF-MTs. A line between dots in the same circle shows intra-dataset links, while a line between dots in different circles corresponds to inter-dataset links. In numbers, there is only one RDF-MT in ChEBI, 234 in DBpedia, six in Drugbank, one in Geonames, 11 in Jamendo, four in KEGG, 53 in LinkedMDB, two in NYTimes, and 80 in SWDF dataset. Four of these RDF-MTs belong to at least two FedBench datasets, modeled as separate circular dots.

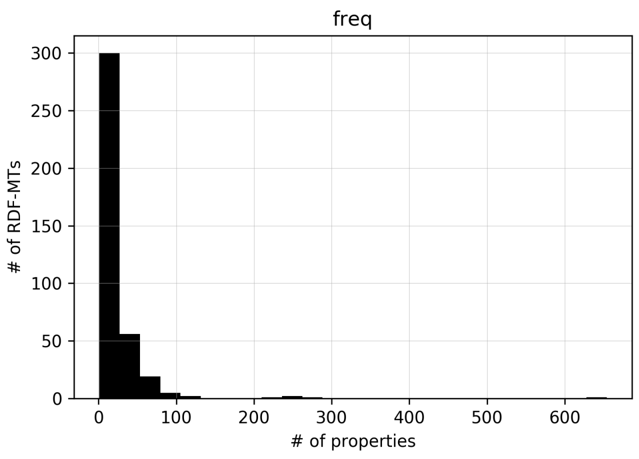


Figure 4.11: **Frequency of FedBench RDF-MTs Per Number of Properties**. Majority of Molecule Templates contain from one to 20 properties.

are generated. While the majority of the RDF-MTs (230) are related to a single dataset, quite a few (4) are shared between two or more datasets. Most of the RDF-MTs have between three and 20 properties, as can be seen in the histogram of Figure 4.11.

From the reported analysis, it can be observed that RDF-MTs can be used to describe characteristics of datasets in terms of connectivity between RDF types represented in each dataset with other datasets

Num of nodes	396
Num of edges	6,317
Graph density	0.081
Avg. num of neighbors	31.904
Connected components	9
Avg. node connectivity	10.624
Transitivity	0.395
Clustering coefficient	0.602

Table 4.3: **FedBench RDF-MT Graph Metrics**. Clustering coefficient (0.602) suggests high number of intra- & inter-dataset links.

in the federation. This answers **Q1** positively in a sense that datasets can be characterized not only by ontology types (RDF types) and predicates, but also using the characteristics of the network between ontology types within the same dataset and with other datasets in a federation.

4.3.2 Comparison of Source Descriptions for Query Processing

We study the impact of RDF-MT on query processing, and compare the effect of computing molecule templates using two existing graph partitioning methods: METIS and SemEP. We name FED-SemEP and FED-METIS, the version of Federated query engine where molecule templates have been computed using SemEP and METIS, respectively. Co-occurrences of predicates in the RDF triples of a dataset *D* are computed. Given predicates *p* and *q* in *D*, co-occurrence of *p* and *q* (*co(p,q,D)*) is defined as follows:

$$co(p,q,D) = \frac{|subject(p,D) \cap subject(q,D)|}{|subject(p,D) \cup subject(q,D)|} \tag{4.1}$$

Where *subject(q,D)* corresponds to the set of different subjects of *q* in *D*. A graph GP_D where nodes correspond to predicates of *D* and edges are annotated with co-occurrence values is created, and given as input to SemEP and METIS. The number of communities determined by SemEP is used to create the corresponding partitions for METIS. METIS- and SemEP-based molecule templates are composed of predicates with similar co-occurrence values. Each predicate is assigned to only one community. For this experiment, we use the following metrics: *i*) *Execution Time*: Elapsed time between the submission of a query to an engine and the delivery of the answers. Time corresponds to absolute wall-clock system time as reported by the Python `time.time()` function. Timeout is set to 300 seconds. *ii*) *Cardinality*: Number of answers returned by the query.

We employ the decomposition and source selection technique described in the next chapter on top of ANAPSID query engine and compare the effect of source descriptions. Figure 4.12 reports on execution time and answer cardinality of the BSBM queries. The observed results suggest that knowledge encoded in RDF-MTs allows FED-RDF-MT to identify query decomposition that speed up query processing by up to two orders of magnitude, while answer completeness is not affected. Specifically, FED-RDF-MTs is able to place in star-shaped subqueries non-selective triple patterns, while FED-SemEP and FED-METIS group non-selective triple patterns alone in subqueries. Thus, the size of intermediate results is larger in FED-SemEP and FED-METIS plans, impacting execution time. FED-RDF-MTs is able to provide a complete answers for all queries, while FED-METIS fails to provide answers for queries B5, B10, B11, and B12, respectively, for queries B8 and B10 FED-SemEP fails. In terms of execution time, FED-RDF-MTs performs better than both FED-METIS and FED-SemEP on all queries. The observed results allow us to positively answer **Q2**, and conclude that RDF-MTs based source descriptions improve the performance

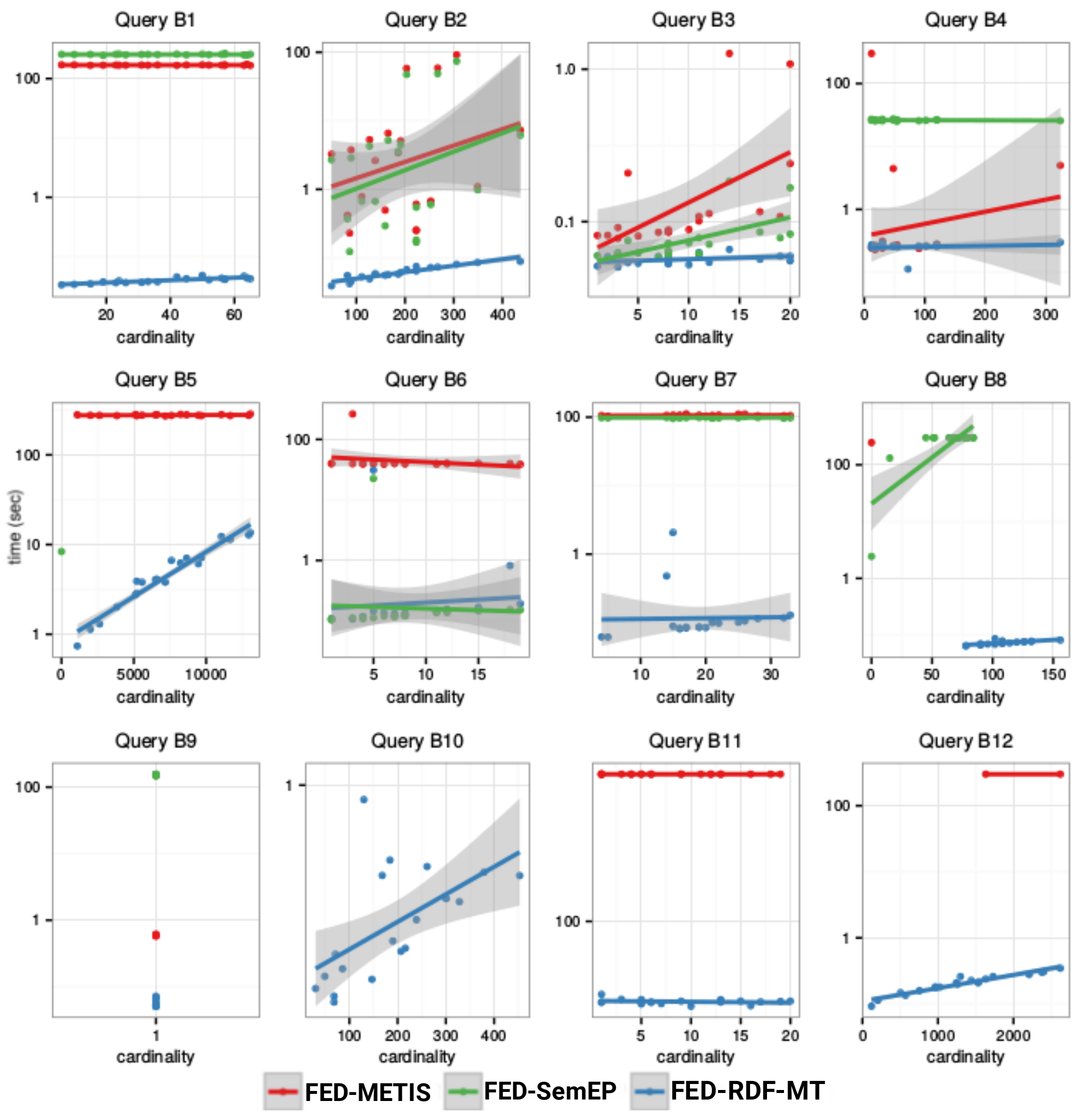


Figure 4.12: **BSBM: Performance of different source descriptions.** RDF molecules are computed using: Algorithm 2, SemEP, and METIS. RDF-MTs allows to identify query decompositions and plans that speed up query processing by up to two orders of magnitude, without affecting completeness.

of query processing, compared to state-of-the-art graph partitioning methods.

4.4 Summary

Data source descriptions provide crucial information that enable federated query processing systems to select relevant sources and optimize queries. RDF molecule templates provide a fine-grained description of heterogeneous data sources as well as connection between them. We analyse the characteristics of three different benchmarks (BSBM, LSLOD, and FedBench) using RDF Molecule Template based descriptions and show that RDF-MTs give not only characteristics of each data sources but also characteristics of the federation as a whole. Apart from out analysis, our experiment showed that logical partitioning of the RDF data using RDF-MTs is more effective and efficient than using state-of-the-art graph partitioning algorithms; SemEP and METIS.

CHAPTER 5

Query Decomposition and Source Selection

Federated query processing systems provide a flexible solution to the problem of query processing over a federation of data sources that are logically integrated as a single data source. This problem has been extensively studied by the database [47, 53, 96–98] and semantic web [34, 49–52, 55] research communities. Chapter 4 discussed the problem of data source description for federation of heterogeneous data sources and propose RDF Molecule Template (RDF-MT) based descriptions. RDF-MT based source description approach is one of the key building block for tackling different sub-problem of federated query processing. In this chapter, we focus on the problem of query processing over a federated of data sources, specifically on the query decomposition and source selection sub-problems. The content of this chapter is based on the publications [89, 90]. The result of this chapter provides an answer to the following research question:

RQ2: How can features represent in data source descriptions be employed to guide the query processing over heterogeneous data sources?

To answer this research question we focus on the decomposition and source selection sub-problem of federated query processing. One of the major sub-problem of federated query processing engines is the selection of the minimal number of relevant sources that can contribute the data required to answer the query completely, as the number of potentially relevant data sources for a query can be very large. Current approaches resort to different source description approaches for identifying the relevant sources of a query. However, the majority of existing approaches only collect coarse-grained source descriptions, e.g., vocabularies or schema utilized in the dataset for modeling the data, and ignore fine-grained characteristics, e.g., classes, properties, and relations. Nevertheless, we deem that fine-grained source descriptions represent building blocks not only for effectively selecting relevant sources, but also for identifying query execution plans that collect the query answers efficiently. We proposed a query decomposition and source selection technique that utilize RDF-MT based source descriptions. We compare the proposed technique with state-of-the-art federated query processing engines over RDF data sources. Figure 5.1 shows the challenge tackled in this chapter with respect to the problem defined in this thesis and the contribution of this chapter.

The contributions in this chapter can be summarized as follows:

- A thorough formalization accompanied by an implementation for federated query processing employing RDF Molecule Templates (RDF-MTs) for selecting relevant sources, query decomposition, and execution.

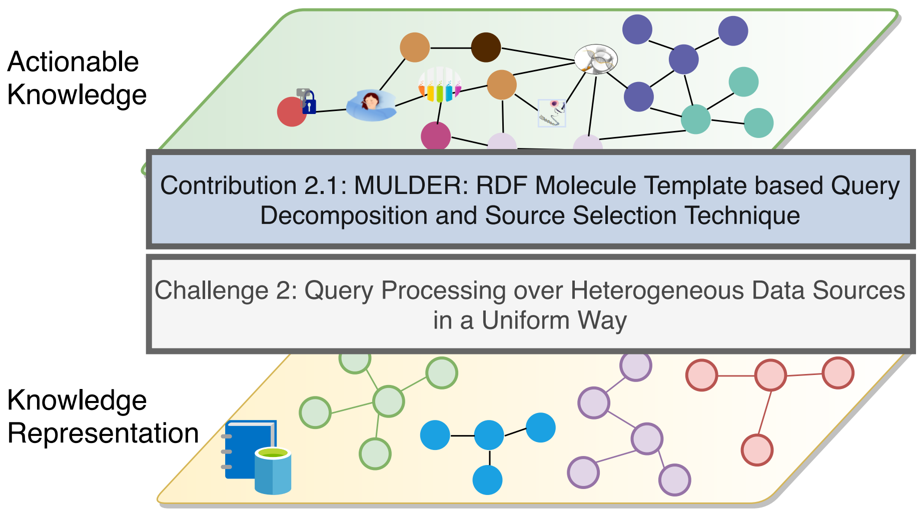

Actionable Knowledge
Contribution 2.1: MULDER: RDF Molecule Template based Query Decomposition and Source Selection Technique
Challenge 2: Query Processing over Heterogeneous Data Sources in a Uniform Way
Knowledge Representation

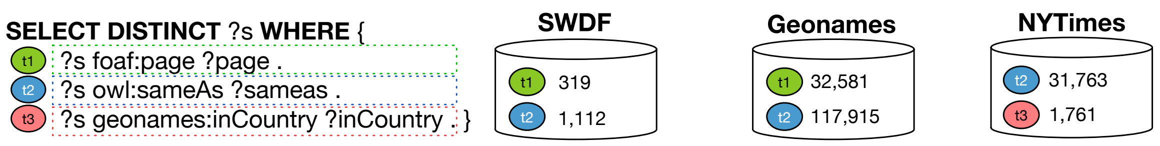
Figure 5.1: **Challenges and Contributions**.This chapter focuses on the problem of describing heterogeneous data sources for data integration, and propose RDF Molecule Templates based data source description for federated query processing

- A federated query processing engine, MULDER, that is able to utilize RDF-MT based descriptions to maximized answer completeness and minimize query execution time.
- An empirical evaluation assessing the performance of RDF-MT based query decomposition and source selection, i.e., the MULDER approach, in terms of query execution time and query answer completeness. The reported results provide evidence that the MULDER approach is able to speed up query execution and enhance answer completeness with respect to the state-of-the-art.
- An experimental study of the continuous efficiency of MULDER in terms of novel metrics are reported; observed results suggest that MULDER performance increases gradually and is competitive to the state-of-the-art adaptive federated query engine ANAPSID [34].

This chapter is structured as follows: first, we motivate the problem of query decomposition and data source selection against a set of data sources in Section 5.1. To address research question **RQ2**, we devise MULDER, a federated query processing engine that utilize RDF-MT based description for decomposition and source selection as well as query plan generation over a federation of RDF data sources. In Section 5.2, we formally define the problem of query decomposition and execution over a set of data sources in a federation. Our technique is based on basic graph patterns (BGPs) of a SPARQL query composed of star-shaped subqueries (SSQs). Then, we present the proposed solution, the MULDER query engine, including algorithms and architecture in Section 5.3. Section 5.4 presents a thorough evaluation of the proposed approach. Finally, Section 5.5 presents the closing remarks of this chapter.

5.1 Motivating Example

We motivate our work by comparing the performance of state-of-the-art federated SPARQL query engines on a federation of RDF data sources from the *FedBench* benchmark [94]. FedBench is a benchmark for evaluating federated query processing approaches (described in section 4.3.1). Although datasets in this benchmark are from different domains, some RDF vocabularies are utilized in more than one dataset. For instance, `foaf` properties are used in DBpedia, GeoNames, SWDF, LinkedMDB, and NYTimes, while RDF triples with the `owl:sameAs` property are present in all the FedBench datasets. Federated



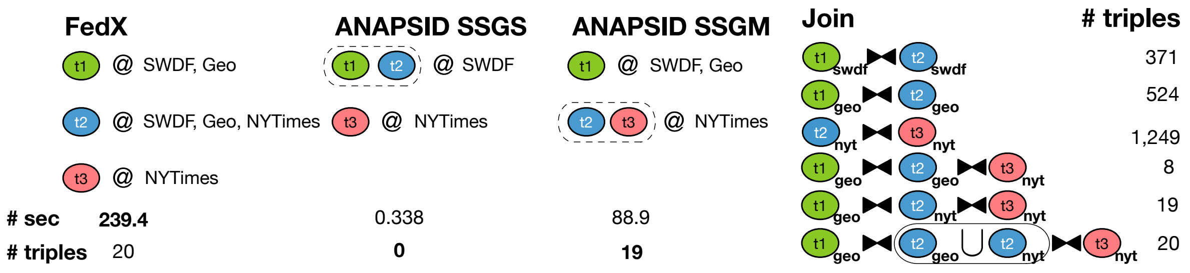
(a) SPARQL Query (b) Relevant FedBench RDF Data Sources

Figure 5.2: **Motivating Example.** (a) SPARQL query over FedBench RDF data sources. (b) FedBench data sources able to execute the query triple patterns. Each triple pattern can be executed in more than one RDF data source.

SPARQL query engines, e.g., ANAPSID [34] and FedX [52], provide a unified view of the federation of datasets, and support query processing over this unified view.

Figure 5.2a presents a SPARQL query on a federation of three data sources: SWDF, Geonmaes, and NYTimes. The query comprises three triple patterns: `t1` can be answered on SWDF and Geonames; NYTimes can answer `t3`, while `t2` can be executed over SWDF, Geonames, and NYTimes respectively. Figure 5.2b reports on the number of answers of `t1`, `t2`, and `t3` over SWDF, Geonames, and NYTimes. Federated query engines rely on source descriptions to select relevant sources for a query. For instance, based on the vocabulary properties utilized in each of the data sources, ANAPSID decides that SWDF, Geonames, and NYTimes are the relevant sources, while FedX contacts each of the federation SPARQL endpoints to determine where `t1`, `t2`, and `t3` will be executed. Furthermore, different criteria are followed to decompose the query into the subqueries that will be posed over the relevant sources to collect the data required to answer the query. As presented in Figure 5.3a, FedX identifies that `t3` composes an exclusive group and can be executed over NYTimes; while `t1` is executed over SWDF and Geonames, and `t2` on all the three datasets. Thus, FedX produces a complete answer by joining the results obtained from executing these three subqueries. Nevertheless, FedX requires 239.4 secs. to execute the query. ANAPSID offers two query decomposition methods: SSGS and SSGM (Figure 5.3a). ANAPSID SSGS only selects one relevant source per triple pattern; execution time is reduced to 0.338 secs., but sources are erroneously selected and the query execution produces empty results. Finally, ANAPSID SSGM builds a star-shaped subquery that includes `t2` and `t3`. The star-shaped subquery is executed on NYTimes, while `t1` is posed over SWDF and Geonames. Execution time is reduced, but only 19 answers are produced, i.e., results are incomplete.

Based on the values of join cardinality reported in Figure 5.3b, the decomposition that produces all



(a) Query Decompositions (b) Join Cardinality

Figure 5.3: **Motivating Example.** (a) Query Decompositions by FedX and ANAPSID. (b) Cardinality of Joins of triple patterns over relevant RDF data sources. FedX decomposition produces complete answers, but at the cost of execution time. ANAPSID decompositions run faster, but produce incomplete results.

the results requires that `t2` is executed over NYTimes and Geonames, while `t1` and `t3` should be only executed in Geonames and NYTimes, respectively. However, because of the lack of source description, neither FedX nor ANAPSID is capable of finding this decomposition. On the one hand, to ensure completeness, FedX selects irrelevant sources for `t1` and `t2`, negatively impacting execution time. On the other hand, ANAPSID SSGS blindly prunes the relevant sources for `t1` and `t2`, and does not collect data from Geonames and NYTimes required to answer the query. Similarly, ANAPSID SSGM prunes Geonames from `t2`, while it is unable to decide irrelevancy of Geonames in `t1`.

5.2 Problem Statement and Proposed Solution

In this section, we define the problem of query decomposition and execution around basic graph patterns (BGPs) of a given SPARQL query Q posed over a federation of data sources.

Given a query Q and a set of data sources, D, in a federation $\mathbb{F}$, the problem of federated query processing is to find a plan that is correct (effective) and efficient. A plan is effective if the results produced by the plan is the same as results that are generated by running the original query Q over union of data sources (as a single dataset). A plan is efficient if the cost of running the plan is minimized. To tackle the problem of federated query processing over a set of data sources in a Semantic Data Lake, we propose a query decomposition and source selection technique, MULDER, that utilized RDF-MT based source descriptions (Chapter 4).

5.2.1 Problem Statement

Our decomposition and source selection technique is based on basic graph patterns (BGPs) and star-shaped subqueries (SSQs), defined in Chapter 2. In this section, we formalize the query decomposition and execution problem over a federation of data sources.

Definition 12 (Query Decomposition) *Given a basic graph pattern BGP of triple patterns $\{t_1,\dots,t_n\}$ and datasets $D=\{D_1,\dots,D_m\}$, a decomposition P of BGP in D, $\gamma(P|BGP,D)$, is a set of service graph patterns SGP=(SQ,SD), where SQ is a subset of triple patterns in BGP and SD is a subset of D.*

Definition 13 (Query Execution over a Decomposition) *The evaluation of $\gamma(P|BGP,D)$ in D, $[[\gamma(P|BGP,D)]]_D$, is defined as the join of the results of evaluating SQ over RDF datasets D_i in SD:*

$$[[\gamma(P|BGP,D)]]_D = JOIN_{(SQ,SD)\in\gamma(P|BGP,D)}(UNION_{D_i\in SD}[[SQ]]_{D_i}) \tag{5.1}$$

After we defined what a decomposition of a query is and how such a decomposed query can be evaluated, we can define the problem of finding a suitable decomposition for a query and a given set of data sources.

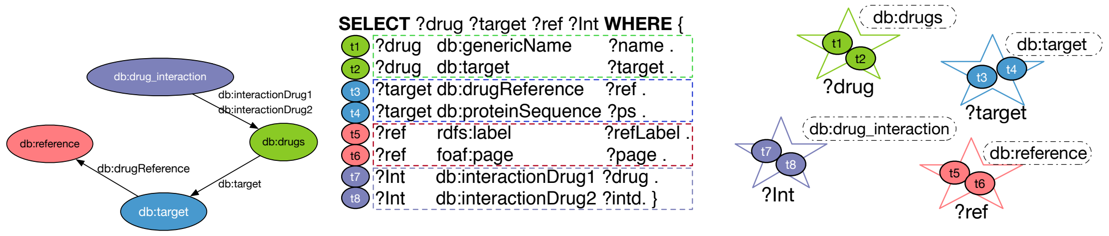
Definition 14 (Query Decomposition Problem) *Given a SPARQL query Q and RDF datasets $D=\{D_1,\dots,D_m\}$, the problem of decomposing Q in D is defined as follows. For all BGPs, $BGP=\{t_1,\dots,t_n\}$ in Q, find a query decomposition $\gamma(P|BGP,D)$ that satisfies the following conditions:*

- *The evaluation of $\gamma(P|BGP,D)$ in D is complete, i.e., if D^* represents the union of datasets in D, then the results of evaluating BGP in D^* and the results of evaluating decomposition $\gamma(P|BGP,D)$ in D are the same, i.e.,*

$$[[BGP]]_{D*} = [[\gamma(P|BGP,D)]]_D \tag{5.2}$$

- *$\gamma(P|BGP, D)$ has the minimal execution cost, i.e., if $cost(\gamma(P'|BGP, D))$ represents the execution time of a decomposition P' of BGP in D, then*

$$\gamma(P|BGP, D) = \underset{\gamma(P'|BGP,D)}{\operatorname{argmin}} \; cost(\gamma(P'|BGP, D)) \tag{5.3}$$



```
SELECT ?drug ?target ?ref ?Int WHERE {
t1 ?drug   db:genericName       ?name .
t2 ?drug   db:target            ?target .
t3 ?target db:drugReference     ?ref .
t4 ?target db:proteinSequence ?ps .
t5 ?ref    rdfs:label           ?refLabel .
t6 ?ref    foaf:page            ?page .
t7 ?Int    db:interactionDrug1 ?drug .
t8 ?Int    db:interactionDrug2 ?intd. }
```

(a) RDF-MTs (b) SPARQL Query (c) Star-shaped Subqueries

Figure 5.4: **Query Decomposition**. (a) RDF-MTs about `db:drug_interaction`, `db:drugs`, `db:target`, `db:reference`. (b) SPARQL query composed of eight triple patterns that can be decomposed into four star-shaped subqueries. (c) Four star-shaped subqueries associated with four RDF-MTs in (a).

5.2.2 Proposed Solution

To solve the query decomposing problem, we devise MULDER, a federated query engine for RDF datasets accessible through Web access interfaces, e.g., SPARQL endpoints. The MULDER *Decomposition & Source Selection* creates a query decomposition with service graph patterns (SGPs) of star-shaped subqueries built according to RDF molecule template (RDF-MT) based data source descriptions. Once the star-shaped subqueries (SSQs) are identified, a bushy plan is built by the MULDER *Query Planning*; where the plan leaves correspond to star-shaped subqueries.

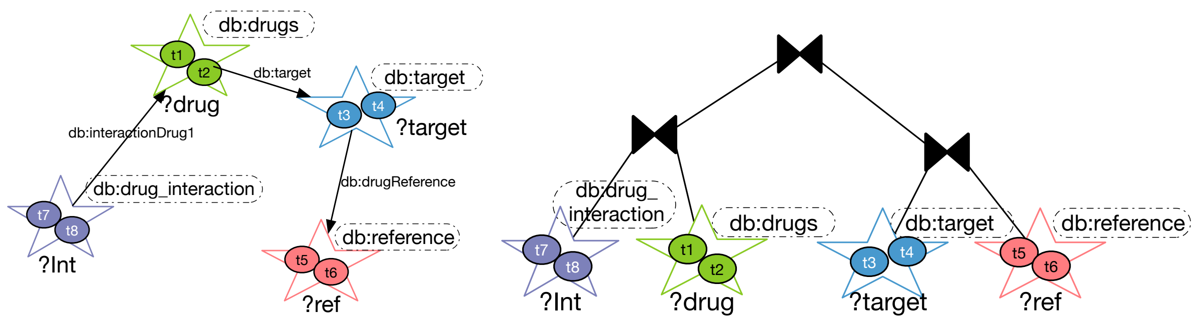
5.3 MULDER: A Federated Query Processing Engine

In this section, we first present the MULDER source selection and query decomposition technique and plan generation approach. Then, we present the MULDER architecture as a federated query processing engine over RDF data sources.

5.3.1 Source Selection and Query Decomposition Technique

Given a SPARQL query MULDER parses the query into star-shaped subqueries and create a *query-graph* where nodes are star-shaped subqueries and edges are join variables. Using RDF-MT based source description, for each node in the *query-graph*, MULDER selects RDF-MT(s) that contain all predicates of a star-shaped subquery. Finally, MULDER selects a source(s) for each star-shaped subquery, if the source contains an RDF-MT(s) with matching properties in a star-shaped subquery, MULDER applies pruning using the actual links that are known between RDF-MTs. MULDER combines triple patterns in a star-shaped subquery as a single SPARQL query to be sent to a single source.

Figure 5.4 shows an example of query decomposition and source selection. The example query in Figure5.4b, contains eight triple patterns. The first step of query decomposition is to identify the star-shaped subqueries (SSQ). In our example, four subqueries which contain two triples patterns each,



(a) Graphs of Joinable SSQs (b) Bushy Plan of Joinable SSQs

Figure 5.5: **Query Planning**. (a) Joinable Graph of Star-shaped Subqueries (SSQs) represents joins between SSQs. (b) Bushy plan of joinable Star-shaped Subqueries (SSQs). Graph of Joinable SSQs is utilized by MULDER Query Planner to create a bushy plan of SSQs where joins between SSQs are maximized.

Algorithm 3 Molecule template based SPARQL query decomposition: *BGP*: Basic Graph Pattern, *WIT*: set of RDF-MTs

```
 1: procedure DECOMPOSE($BGP, WIT$)
 2:     $CM \leftarrow \{\}$                                          ▷ $CM$ - Candidate RDF-MTs
 3:     $SSQs = getStarShapedSubqueries(BGP)$                        ▷ Subject stars
 4:     for $s \in SSQs$ do
 5:         for $RDF_{MT} \in WIT$ do
 6:             if $predicatesIn(s) \subseteq predicatesIn(RDF_{MT})$ then
 7:                 $CM[s].append(RDF_{MT})$
 8:             end if
 9:         end for
10:     end for
11:     $JSSQ = getJSSQs(SSQs)$                                      ▷ Query-graph of Joinable SSQs
12:     $conn = connectedRDFMTs(SSQs, JSSQ, CM)$                     ▷ selected RDF-MTs graph
13:     $DQ = prune(SSQs, JSSQ, conn)$
14:     return $DQ$                                                  ▷ decomposed query
15: end procedure
```

are identified, i.e., `?drug` (t_1, t_2), `?target` (t_3, t_4), `?ref` (t_5, t_6), and `?Int` (t_7, t_8), named after the shared subject variable of star-shaped subqueries. Each of SSQs are then associated with RDF-MTs that contain predicates in SSQs, as shown in Figure 5.4c.

The MULDER query decomposer is sketched in Algorithm 3. Given a BGP and a set of RDF-MTs (WIT), SSQs are first identified (Line 3). Then, RDF-MTs which contain all predicates in SSQ are determined from WIT as candidate RDF-MTs (Line 4–10). Furthermore, linked candidate RDF-MTs with respect to Joinable SSQs (Line 11) are identified (Line 12). Finally, candidate RDF-MTs are pruned, i.e., candidate RDF-MTs that contain all predicates in SSQ but are not linked to any RDF-MT that matches Joinable SSQ are excluded (Line 13). SSQs that have more than one matching RDF-MT from the same Web access interface will be decomposed as one service graph pattern. However, if matching RDF-MTs are from different Web access interfaces, then MULDER decomposes them; the UNION operator is used during query execution to collected the data from each Web access interface.

Figure 5.5a shows joinable star-shaped subqueries (SSQ) that share at least one variable, i.e., `?Int` is joinable with `?drug` via predicate `db:interactionDrug1`, while `?drug` is joinable with

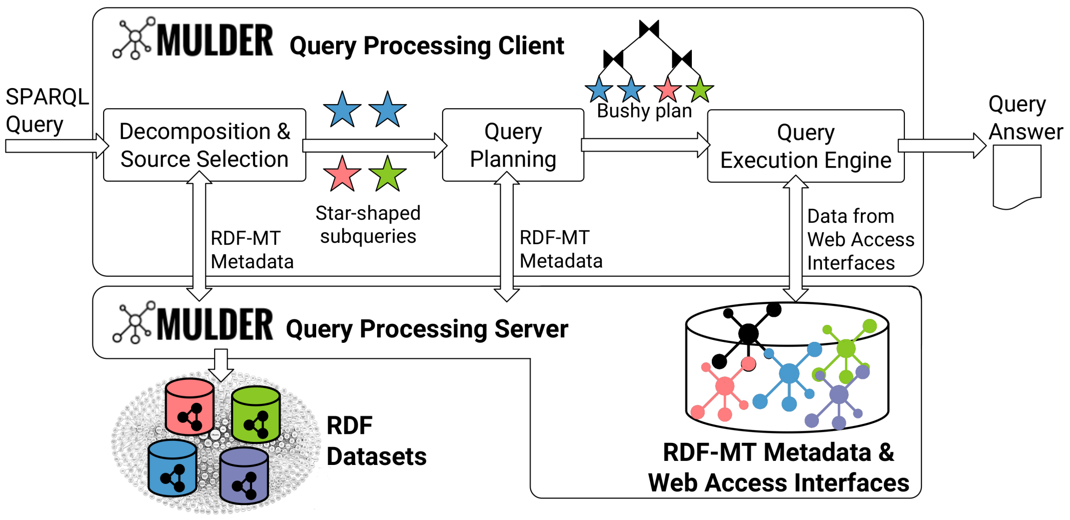

MULDER Query Processing Client
SPARQL Query
Decomposition & Source Selection
Star-shaped subqueries
Query Planning
Bushy plan
Query Execution Engine
Query Answer
RDF-MT Metadata
RDF-MT Metadata
Data from Web Access Interfaces
MULDER Query Processing Server
RDF Datasets
RDF-MT Metadata & Web Access Interfaces

Figure 5.6: **The MULDER Client-Server Architecture**. MULDER query processing client receives SPARQL queries, creates query decompositions with star-shaped subqueries, and identifies and executes bushy plans. MULDER query processing server collects both RDF-MT metadata about RDF datasets and results of executing queries over Web access interfaces, e.g., SPARQL endpoints.

`?target` via `db:target`. Furthermore, `?target` is joinable with `?ref` through `db:drugReference` property. Finally, MULDER query planner generates bushy plans combining SSQs (Figure 5.5b). The problem of identifying a bushy plan from conjunctive queries is known to be NP-complete [99]. MULDER planner implements a greedy heuristics based approach to generate a bushy plan, where the leaves correspond to SSQs, and the number of joins between SSQs is maximized while the plan height is minimized. The heuristics used by MULDER planner are discussed in section 6.3.1.

5.3.2 The MULDER Architecture

The MULDER architecture is depicted in FigureFigure 5.6. The MULDER *Query Processing Client* receives a SPARQL query, decomposes it, performs source selection based on RDF-MT metadata, and, afterwards, identifies a bushy plan against the selected RDF datasets. The MULDER *Query Engine* executes the bushy plan and contacts the MULDER query processing server to evaluate Service Graph Patterns over the Web access interfaces. Further, the MULDER *Query Processing Server* receives requests from the MULDER client to retrieve RDF-MT metadata about RDF datasets, e.g., metadata about properties of RDF molecules contained in these RDF datasets.

5.4 Empirical Evaluation

We empirically study the efficiency and effectiveness of MULDER query decomposition and source selection techniques. We compare MULDER with the federated query engines ANAPSID and FedX for three well-established benchmarks – BSBM, FedBench, and LSLOD. Furthermore, we evaluate the *continuous efficiency* of MULDER compared to ANAPSID. The following research questions are evaluated: **Q1)** Is the effectiveness and efficiency of the query processing process impacted by the MULDER query decomposition and source selection technique? **Q2)** Is the continuous efficiency of the answer generation process impacted by the MULDER query decomposition and source selection technique?

Query	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
#BGPs	1	1	2	5	1	2	1	1	1	1	1	1
#TP	4	6	12	17	8	10	8	4	5	11	25	21
#SSQs	1	1	2	7	2	4	2	2	2	3	4	4
UNION			X									
OPTIONAL				X		X						
DISTINCT	X		X		X		X		X	X	X	X

Table 5.1: **BSBM queries characteristic**.

Query	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
#BGPs	2	1	1	1	2	1	1	1	1	1
#TP	4	7	6	5	5	3	4	3	8	8
#SSQs	2	3	4	2	3	2	3	2	2	2
UNION										
OPTIONAL					X					
DISTINCT									X	X

Table 5.2: **LSLOD queries characteristic**.

The MULDER[1] decomposition and source selection, and query planning components are implemented in Python 2.7.10. MULDER plans are executed using ANAPSID [34] physical operators. Experiments are executed on two Dell PowerEdge R805 servers, AMD Opteron 2.4GHz CPU, 32 cores, 256GB RAM. BSBM (The Berlin SPARQL Benchmark), FedBench, and LSLOD datasets are deployed on one machine as SPARQL endpoints using *Virtuoso 6.01.3127*, where each dataset resides in a dedicated Virtuoso docker container.

Benchmarks

Three benchmarks are utilized to assess our research questions: i) BSBM - The Berlin SPARQL Benchmark; ii) FedBench; and iii) LSLOD.

BSBM - The Berlin SPARQL Benchmark. The Berlin SPARQL Benchmark [92] is a synthetic dataset focusing on an e-commerce use case where a set of products are offered by different vendors and consumers and reviewers have posted reviews about these products on various review sites. We use BSBM to generate 12 SELECT queries (with 20 instantiations each) over a generated dataset containing 200 million triples[2]; characteristics of these queries are presented in Table 5.1. We partitioned the dataset using *rdf:type* classes and created eight SPARQL endpoints, one per each class and one endpoint which contains the whole dataset.

FedBench. FedBench [94] is a benchmark suite for analyzing both the efficiency and effectiveness of federated query processing strategies for different use cases on semantic data. We run 25 FedBench queries[3], including cross-domain queries (CD), linked data queries (LD), and life science queries (LS). Additionally, 10 complex queries (C) proposed by [27] are considered. The queries are executed against the FedBench datasets from **cross-domain** and **life-science** collections. A SPARQL endpoint able to

[1] `https://github.com/SDM-TIB/Mulder`
[2] BSBM queries can be found in the Appendix B.1
[3] FedBench queries can be found in `http://fedbench.fluidops.net/resource/Queries`

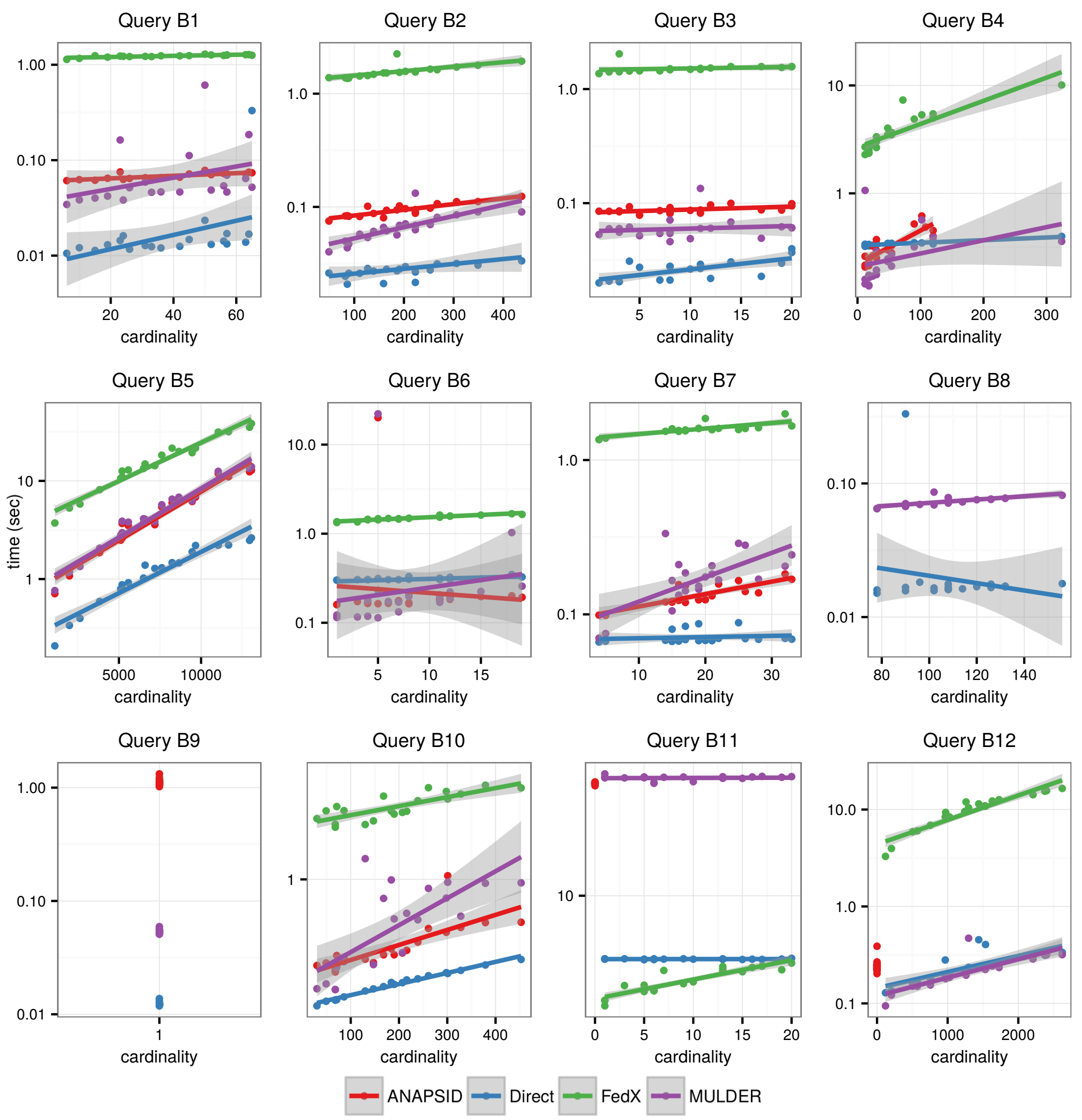


Figure 5.7: **BSBM: Performance of Federated Engines.** MULDER and ANAPSID outperform FedX in terms of query execution time, while MULDER overcomes ANAPSID in terms of completeness. Direct represents a unified SPARQL endpoint over one dataset with all the federation RDF triples.

access a unified view of all the FedBench datasets (i.e., the RDF dataset D^* in Equation 5.2) serves as gold standard and baseline.

LSLOD. LSLOD [93] is a benchmark composed of 10 real-world datasets of the Linked Open Data (LOD) cloud from life sciences domain. We run 10 simple queries[4] provided for LSLOD datasets in [93], characteristics of these queries are presented in Table 5.2.

5.4.1 Comparison of Federated Query Engines

We evaluate the efficiency and effectiveness (in terms of execution time and answer completeness, respectively) of RDF-MT query processing technique implemented in MULDER compared with the state-of-the-art federated query engines, FedX and ANAPSID. We created a unified view of all datasets

[4] LSLOD queries can be found in Appendix B.1

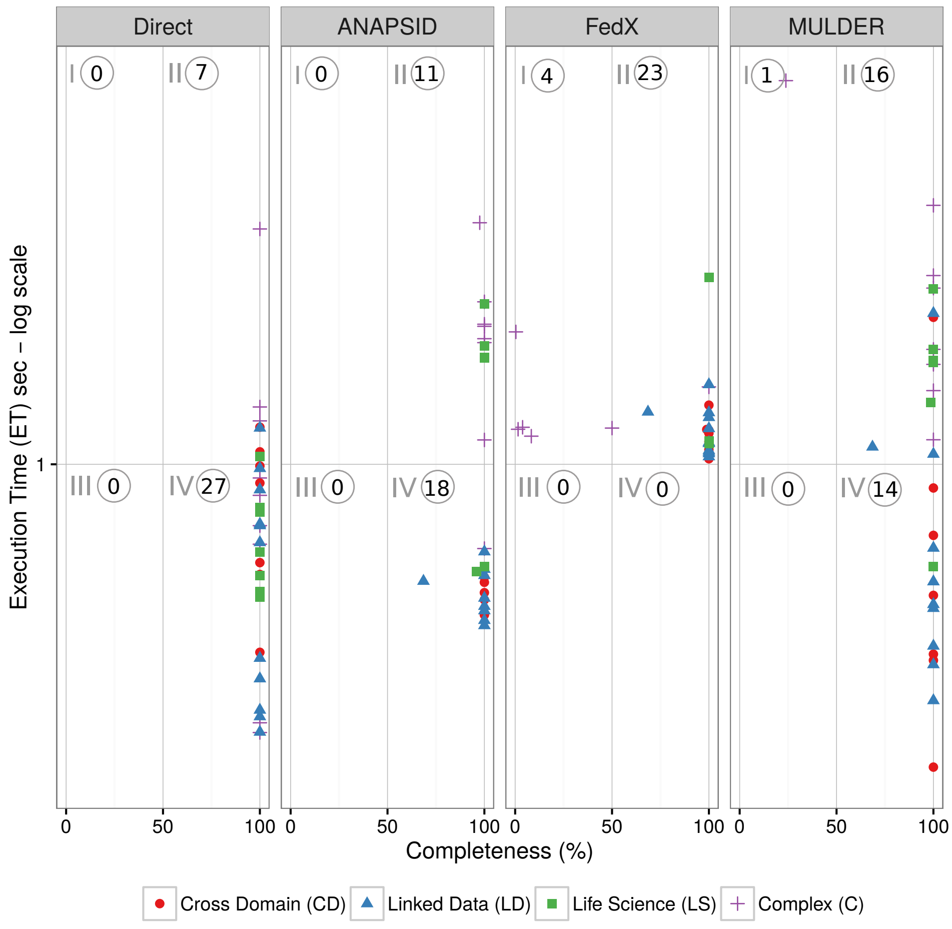


Figure 5.8: **FedBench: Execution Time and Completeness of Federated Engines.** Plots are divided into four quadrants: Incomplete results and slower execution time are reported in Quadrant I; results in Quadrant II correspond to complete results with lower performance; Quadrant III reports faster execution time but incomplete results; Quadrant IV indicates complete results and faster execution time. ANAPSID, FedX and MULDER manage to answer 29, 27, and 31 queries, respectively. Direct represents a unified SPARQL endpoint that is able to answer 34 of the 35 benchmark queries before timing out.

in a benchmark via a *direct* SPARQL endpoint as a baseline. For this experiment, we compare federated query processing techniques using the following metrics: *i*) *Execution Time*: Elapsed time between the submission of a query to an engine and the delivery of the answers. Time corresponds to absolute wall-clock system time as reported by the Python time.time() function. Timeout is set to 300 seconds. *ii*) *Cardinality*: Number of answers returned by the query. *iii*) *Completeness*: Query result percentage with respect to the answers produced by the unified SPARQL endpoint created as the union of all datasets in the benchmark.

Performance of BSBM Queries. Figure 5.7 reports on the throughput of the federated engines ANAPSID, FedX, and MULDER for all BSBM queries. In many queries, MULDER and ANAPSID exhibit similar query execution times. FedX is slower than the two federated engines by at least one order of magnitude. ANAPSID returns query answers fast but at the cost of completeness, as can be observed in the queries B4, B7, B11, and B12. In addition, FedX and ANAPSID fail to answer B8 which

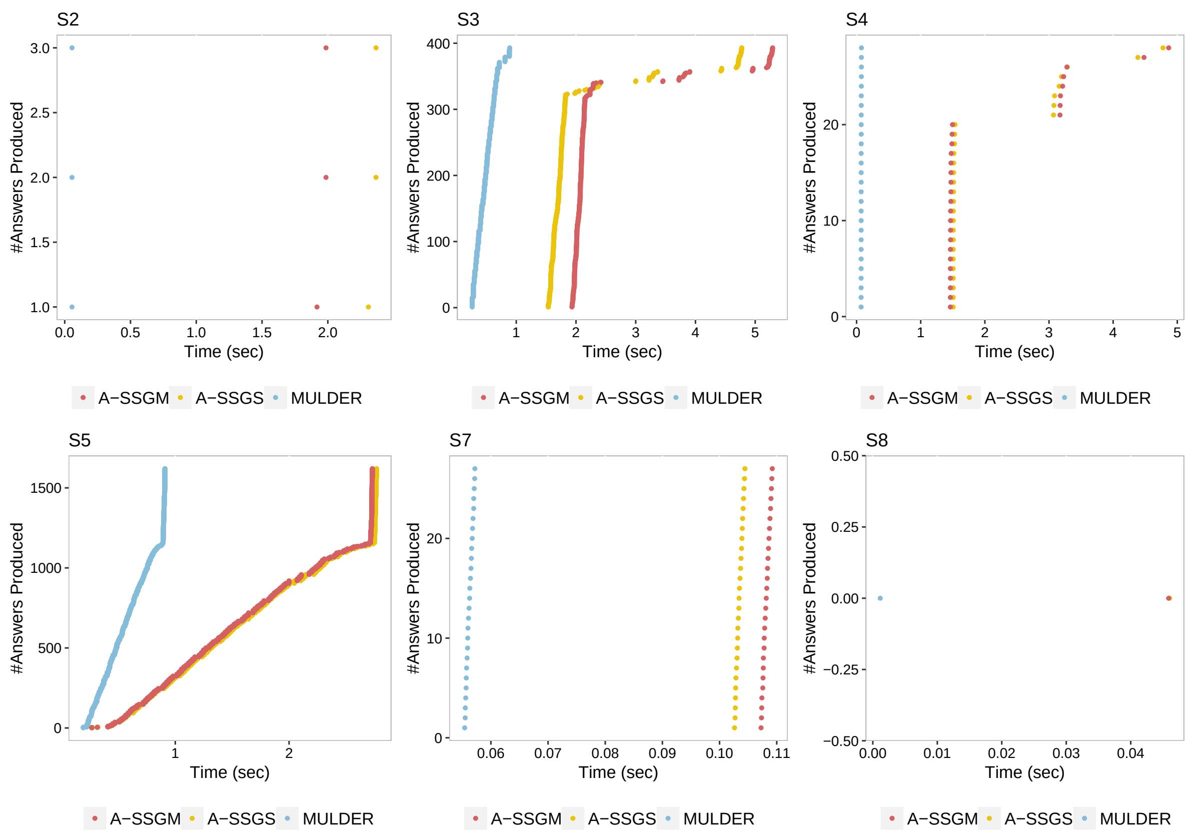


Figure 5.9: **LSLOD: Answer traces.** Continuous query performance. Y-axis shows the number of answers produced, and x-axis shows time in seconds (time.time()).

is completely answered by MULDER.

Performance of FedBench Queries. Figure 5.8 visualizes the results of the four FedBench groups of queries (CD, LD, LS, C) in terms of answer completeness and query execution time. Measurements that are located in Quadrants I and III indicate bad performance and incomplete results, points in Quadrant IV are the best in terms of execution time and completeness, i.e., they correspond to a solution to the query decomposition problem; finally, points in Quadrant II show complete results but slower execution times. MULDER outperforms ANAPSID and FedX with regard to the number of queries it manages to answer: ANAPSID answers 29, FedX 27, and MULDER 31 out of 35 queries (Query C9 could not be answered by any of the engines). In particular, MULDER delivers answers to queries C1, C3, C4, LS4, LS5, and LS6 for which FedX fails and CD6 and LD6 for which ANAPSID fails. FedX returns complete and partially complete results for 20 and 7 queries respectively, exhibiting high execution times though (>1s). In comparison to ANAPSID, MULDER achieves in general higher completeness of results, but at the cost of query execution time. For instance, C2, C8, and LD1 are answered by ANAPSID faster by almost one order of magnitude. Results observed in both benchmarks, i.e., BSBM and FedBench, allow us to positively answer **Q1** and **Q2**, and conclude that RDF-MTs enable MULDER decomposition and planning methods to identify efficient and effective query plans.

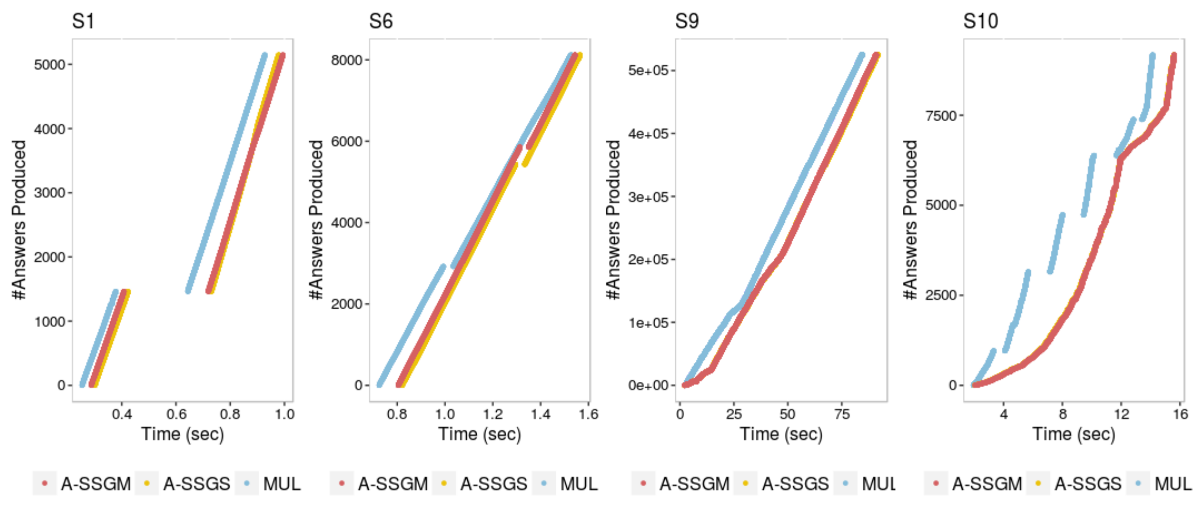


Figure 5.10: **LSLOD: Answer traces.** continuous query performance. y-axis shows the number of answers produced, and x-axis show time in seconds (time.time()).

5.4.2 Measuring Continuous Efficiency of MULDER

In this experiment, we evaluate the efficiency of MULDER in terms of continuous generation of answers of a query. A continous efficiency (diefficiency) of query engine is measured using metrics proposed in [100]. A continuous efficiency (diefficiency) of a query engine can be analyzed from the answer traces. Answer traces are a sequence of pairs (t_i, μ_i) where t_i is the time-stamp that the *ith* answer, μ_i, is produced. Two methods proposed to measure the continuous effect of the engine are $dief@t$ and $dief@k$. Values of these metrics correspond to the number of answers produced in function of time, also known as *Answer Distribution Function*. Diefficiency at time t, $dief@t$, measures the continuous efficiency of an engine in the first t time units of query execution, while diefficiency at k answers, $dief@k$, measures the diefficiency of an engine while producing the first k answers of a query after the first answer is produced.

$Dief@t$ metrics computes AUC (area-under-the-curve) of answer distribution until time t. Given an approach ρ, a query Q, and answer distribution function $X_{\rho,Q}$ while ρ executes Q, $dief@t$ is computed as:

$$dief_{\rho,Q}@t := \int_0^t X_{\rho,Q}(x)dx \tag{5.4}$$

$Dief@k$ metrics computes AUC of answer distribution until the point in time t_k when the engine produces the *kth* answer, as recorded in the answer trace. Given an approach ρ, a query Q, and answer distribution function $X_{\rho,Q}$ while ρ executes Q, $dief@k$ is computed as:

$$dief_{\rho,Q}@k := \int_0^{t_k} X_{\rho,Q}(x)dx \tag{5.5}$$

where $t_k \in \mathbb{R}$ is the point in time when ρ produces the *kth* answer of Q.

Since both ANAPSID and MULDER generate results incrementally, we compared the two heuristics techniques of ANAPSID, i.e., SSGS and SSGM, with MULDER using diefficiency metrics. Figure 5.9 shows answer traces and continuous query performance, of the three approaches for S2, S3, S4, S5, S7, and S8 LSLOD benchmark queries. As shown in Figure 5.9, for these queries MULDER outperforms both ANAPSID heuristics techniques. On S2, the answer trace shows that MULDER produced more results faster in the first 0.05 secs of overall execution than other both ANAPSID approaches. All

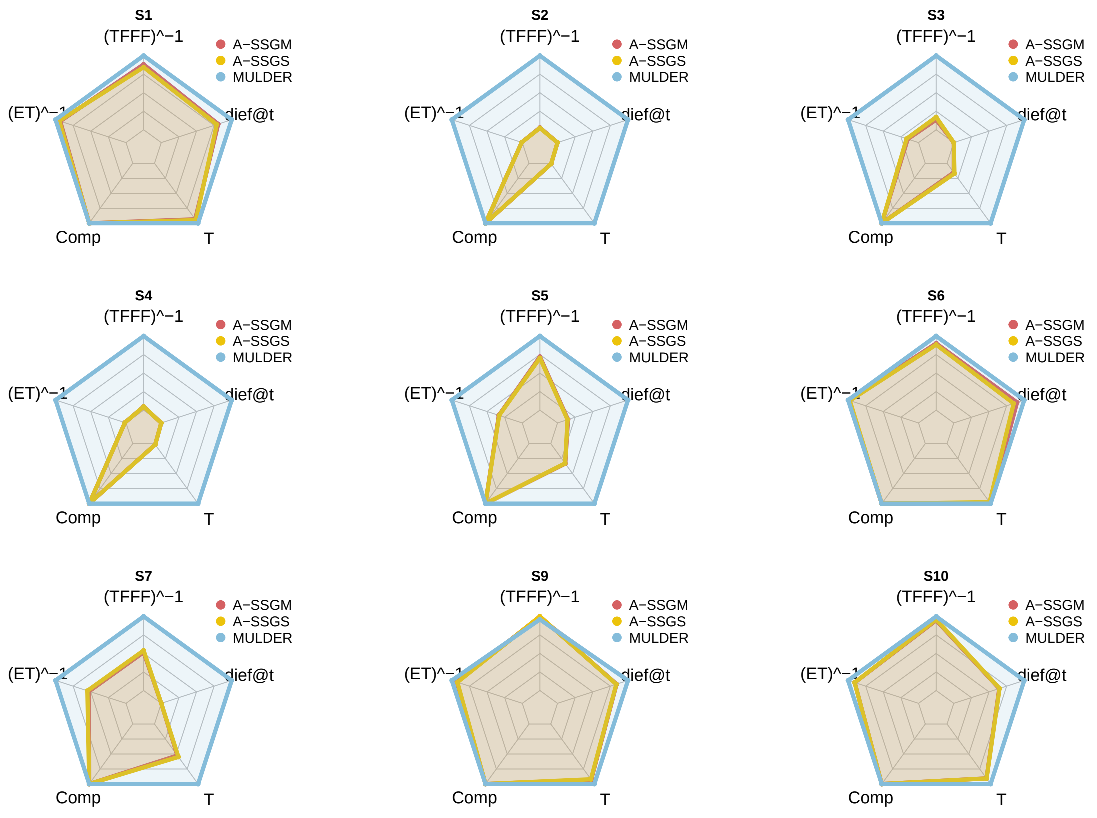


Figure 5.11: **LSLOD: Efficiency and Completeness Metrics.** Performance per LSLOD benchmark query of ANAPSID-SSGS (A-SSGS), ANAPSID-SSGM (A-SSGM) and MULDER query approaches. Axes correspond to: Inverse of Time for the first tuple ($TFFT^{-1}$), Inverse of Total execution time (ET^{-1}), Number of answers produced (Comp), Throughput (T), and $dief@t$. Interpretation of all metrics (axes): 'higher is better'.

approaches return 0 results for query S8, none of the data sources in the benchmark is able to answer the query. Even though, the result is empty, MULDER produces the empty answer earlier than ANAPSID and without contacting any source. Therefore, we excluded query S8 from the reported results on the continuous efficiency of approaches in this section. Figure 5.10 shows answer traces for S1, S6, S9, and S10 LSLOD benchmark queries where all approaches produced answers in a uniform way. On these queries, MULDER produced answers faster than both ANAPSID approaches. For queries S1 and S6, MULDER is able to produce the first answer (row) earlier than ANAPSID-SSGS and ANAPSID-SSGM. For query S6 though, all approaches produce results continuously, MULDER produces slightly higher number of answers faster than others until the first sec. Finally, on query S10, the versions of ANAPSID produced results continuously compared to MULDER. However, MULDER produces more answers faster than other approaches.

In Figure 5.11 reports the performance of approaches using multiple metrics to evaluate the overall performance, completeness and continuous efficiency in time t, i.e., Inverse of Time for the first tuple ($TFFT^{-1}$), Inverse of Total execution time (ET^{-1}), Number of answers produced (Comp), Throughput (T), and $dief@t$, using radar plots. In this plot, the interpretation of the metrics in each axis is **'higher is better'**. As clearly shown, MULDER outperforms ANAPSID-SSGS and ANAPSID-SSGM in almost all metrics and is able to continuously produce results faster. For queries, S1, S6, and S10, all approaches show uniform behavior for all metrics. On the other hand, the performance of both ANAPSID-SSGS

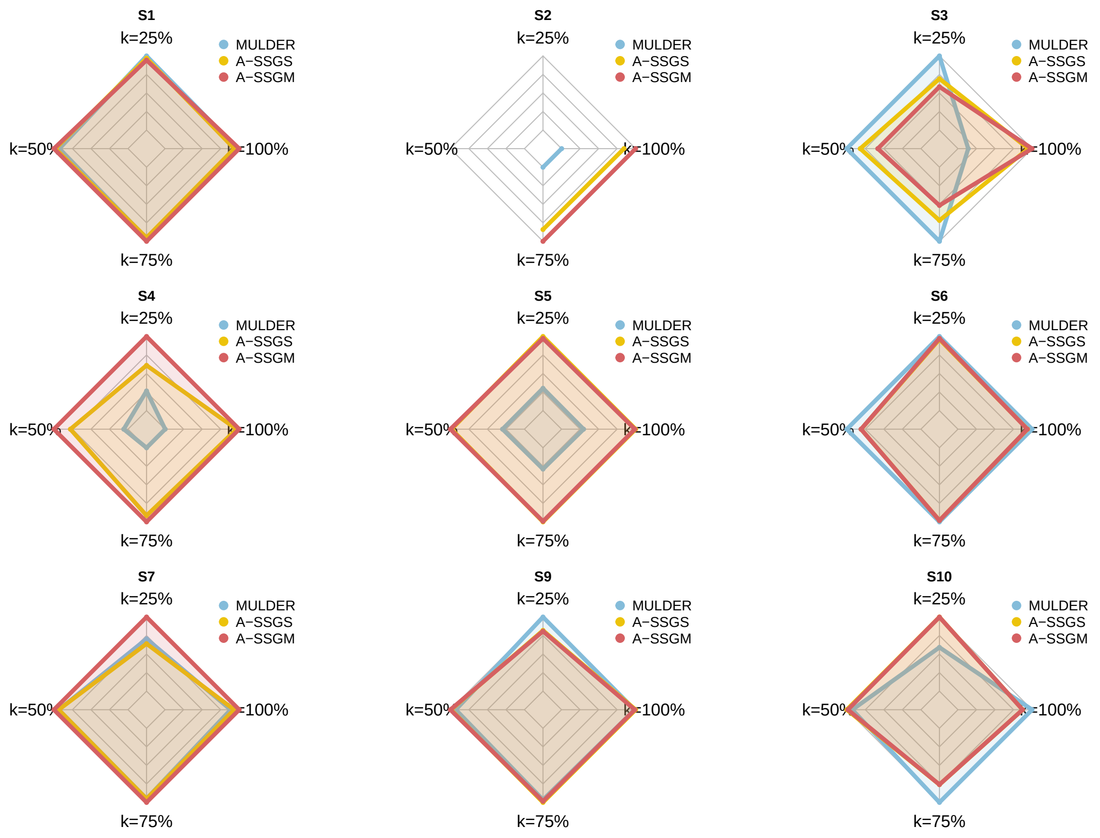


Figure 5.12: **LSLOD: Comparison of Diefficiency.** Diefficiency while producing a portion of k of answers per LSLOD benchmark query of ANAPSID-SSGS (A-SSGS), ANAPSID-SSGM (A-SSGM), and MULDER query approaches. Performance is measured with $dief@k$, with $k = 25\%$, $k = 50\%$, $k = 75\%$, $k = 100\%$. Interpretation of the axes: 'lower is better'.

and ANAPSID-SSGM on queries, S2, S3, S4, S5, and S7 is almost the same. This is because, LSLOD benchmark datasets do not have sources that share the same RDF-MTs; therefore, the source selection of SSGM is mostly the same as SSGS heuristics. MULDER performs better in all metrics.

We analyze the diefficiency (continuous efficiency) achieved by the two heuristics of ANAPSID and MULDER approach while producing the first k results for LSLOD benchmark queries. Figure 5.12 reports on the $dief@k$ values while producing 25%, 50%, 75%, and 100% of the query results. The value of k is selected by taking the minimum number of results returned from the evaluated approaches. In this plot, the interpretation of the metrics is **'lower is better'**. All approaches perform similar efficiency on the first query, S1. On queries S3, S6, S9, and S10 approaches show different behaviour in different values of k. For instance, on S3 MULDER is able to deliver results faster than ANAPSID-SSGS and ANAPSID-SSGM during the last quarter of overall results, but while k is 25%, 50%, and 75% of the overall results both ANAPSID heuristics delivers faster than MULDER. Even though MULDER performed better on $dief@t$ metrics, as shown in Figure 5.11, it produces results slower on k value at 25%, 50%, and 75% on query S3, 50% on S6, 25% on S9, and 75% and 100% of the query results than ANAPSID. On 25%, 75% and 100% of S6 overall query results, 50%, 75%, and 100% of S7 overall query results, 50%, 75%, and 100% of S9 overall query results, and 50% of S10 overall query results all approaches have similar performance of $dief@k$ metrics. For S2, S4, S5, and S7 queries MULDER is able to continuously deliver results faster on all values of k, i.e., 25%, 50%, 75%, and 100% of the overall results and 100% for query

S3. In this experiment, we evaluate the continuous efficiency of MULDER compared to ANAPSID. The observed results for both $dief@t$ and $dief@k$ metrics allow us to positively answer **Q3** and conclude that RDF-MTs based query decomposition and planning techniques implemented in MULDER enable for a continuous performance during the answer generation process.

5.5 Summary

MULDER is federated query processing engine that provide solutions to the problems of source selection, query decomposition and planning, in order to achieve both efficiency in terms of execution time and completeness of results. MULDER resorts to RDF Molecule Templates for describing the structure of RDF data sources and guiding the decomposition of queries. It also provides a query engine for federated access to SPARQL endpoints, able to bridge between parts of a query to be executed in a federated way. We showed through an extensive evaluation using three different benchmarks (BSBM, FedBench, and LSLOD) that MULDER is significantly reducing query execution time and increasing answer completeness in comparison to the state-of-the-art federated engines. In fact, MULDER has comparable results with ANAPSID, however, the latter may deliver more incomplete results. Furthermore, the analysis of the continuous efficiency of MULDER as an adaptive federated engine demonstrated that it is able to continuously deliver results equal or faster than the ANAPSID adaptive query engine.

CHAPTER 6

Query Planning and Optimization

The need for efficient big and heterogeneous data management, and query processing techniques has been gaining attention, as the growing amount of heterogeneous data became available through various platforms [1, 2]. Data Lakes that integrate different data sources need to handle the variety and volume of data efficiently and effectively. Contrary to existing SPARQL federated query engines, federated query processing over Data Lakes demands the integration and semantic description of data collected from heterogeneous data sources. Previous chapters, Chapter 4 and Chapter 5 present, respectively, data source description and query decomposition and source selection sub-problems. Chapter 5, presents the MULDER approach that employs an optimization technique for federated query processing assuming autonomous and homogeneous data sources in a federation. This chapter deals with the query planning and optimization sub-problem of federated query processing considering not only autonomy but also the heterogeneity of data sources in the Semantic Data Lake. The content of this chapter is based on the publication [91]. The result of this chapter aims to answer the following research question at the level of query planning and optimization sub-problem of federated query processing:

RQ2: How can features represent in data source descriptions be employed to guide the query processing over heterogeneous data sources?

To address this research question, we focus on the query plan generation utilizing the capabilities and semantic description of heterogeneous data sources. We propose a set of source specific heuristics that guide the plan generation approach. The proposed optimization techniques considers the RDF-MT based data source descriptions to generate an efficient execution plan. Figure 6.1 shows the challenges tackled in this chapter and the contribution of this chapter.

The contributions of this chapter can be summarized as follows:

- A formalization of federated query processing employing RDF Molecule Templates (RDF-MTs) for selecting relevant sources, query decomposition, planning, and execution.
- A set of source specific heuristics that utilize the available source descriptions to optimize the query execution plan.
- A federated query engine, Ontario, that is able to utilizes RDF-MT based descriptions which maximize answer completeness and minimize query execution time against heterogeneous data sources in a Semantic Data Lake.
- An empirical evaluation assessing the performance of RDF-MT based query decomposition and source selection, i.e., the Ontario approach, in terms of query execution time and query answer

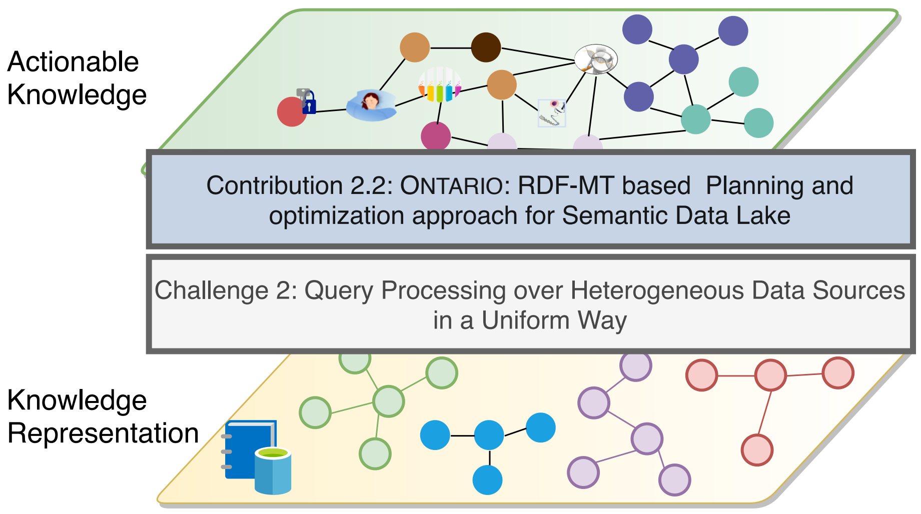

Actionable Knowledge
Contribution 2.2: ONTARIO: RDF-MT based Planning and optimization approach for Semantic Data Lake
Challenge 2: Query Processing over Heterogeneous Data Sources in a Uniform Way
Knowledge Representation

Figure 6.1: **Challenges and Contributions.**This chapter focuses on the problem of federated query processing at the level of query planning and execution over heterogeneous data sources for data integration, and propose Ontario federated query processing against a Semantic Data Lake

completeness. The reported results provide evidence that the Ontario approach is able to speed up query execution and enhance answer completeness with respect to the state-of-the-art.

This chapter is structured as follows: Section 6.1 motivate the problem of plan generation in a federation of heterogeneous data sources. In Section 6.2, we formally define the problem of federated query processing against heterogeneous data sources in Semantic Data Lake. Then, we present the proposed solution, Ontario query processing engine in Section 6.3. Section 6.4 reports the results of an empirical evaluation of the proposed optimization techniques. Finally, Section 6.5 presents the closing marks of this chapter.

6.1 Motivating Example

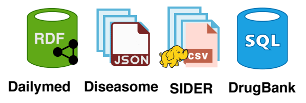
In the biomedical domain, frequently complex queries need to be answered with multiple data sources and data models. Especially in this domain, flexible data management and integration techniques are required due to the variety of tools and formats data is collected, generated, and processed. To provide a unified view over these heterogeneous data sources, mapping rules are utilized to describe the required transformations from raw data into the unified schema. These mappings enable the translation of queries from the unified schema into queries against the sources using native access interfaces.

We motivate our work by comparing the performance of federated SPARQL query engines over a federation of data sources that provide a SPARQL-based access interface. For instance, a SPARQL query in Figure 6.2a requires to collect the name of possible drug targets, chemical formula, and side effects of drugs labeled by FDA that have the active substance Simvastatin. To answer this query, four (4) datasets (Figure 6.2b) need to be accessed. Dailymed[1] publishes FDA label information about marketed drugs in the United States; Diseasome[2] makes availabe a network of disorders and disease genes; DrugBank[3] reports information about drugs and drug targets, and SIDER[4] presents information on drug side effects.

[1] https://dailymed.nlm.nih.gov
[2] https://old.datahub.io/dataset/fu-berlin-diseasome
[3] https://www.drugbank.ca
[4] http://sideeffects.embl.de/

```
SELECT DISTINCT ?drug ?disName ?drugformula ?sename
WHERE {
  t1 ?drug      dailymed:activeIngredient         dailymed:Simvastatin .
  t2 ?drug      dailymed:genericDrug              ?dbdrug .
  t3 ?drug      dailymed:possibleDiseaseTarget    ?disease .
  t4 ?drug      owl:sameAs                        ?sadrug .
  t5 ?disease   rdfs:label                        ?disName
  t6 ?sadrug    sider:sideEffect                  ?seffect .
  t7 ?seffect   sider:sideEffectName              ?sename .
  t8 ?dbdrug    drugbank:chemicalFormula          ?drugformula
}
```

(a) SPARQL Query: Find targets, and side effects of drugs with active ingredient Simvastatin.

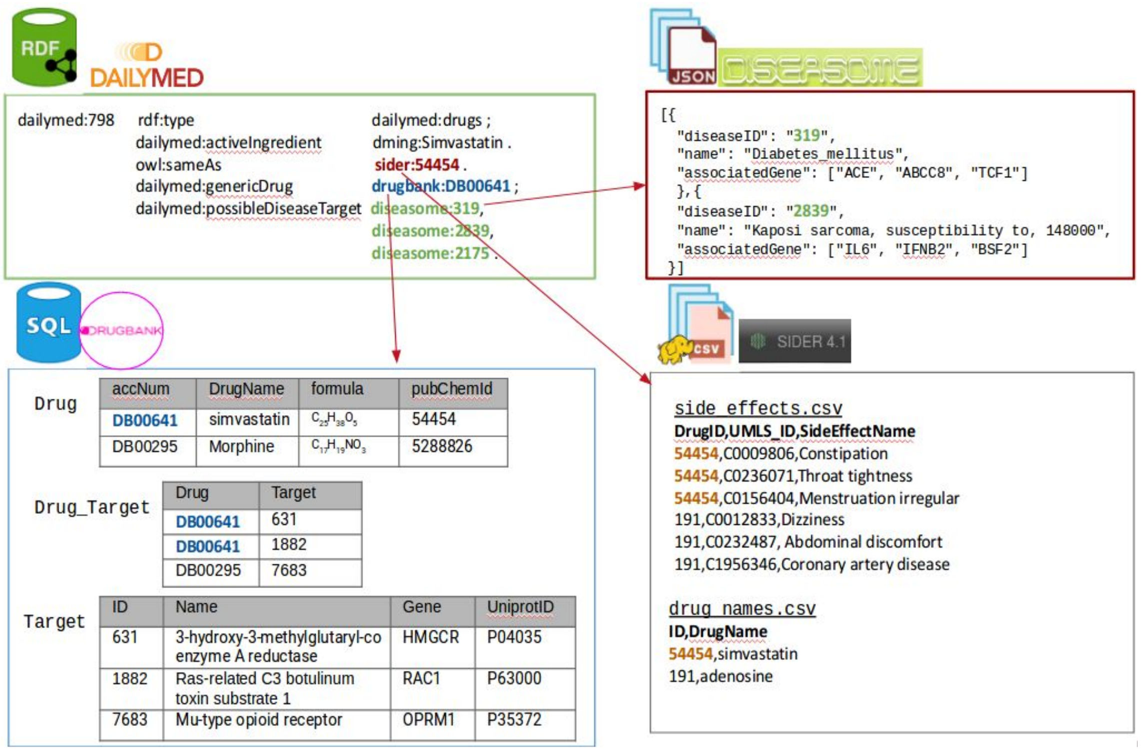


(b) Data Sources in a Data Lake

Figure 6.2: **Motivating Example**. (a) A SPARQL query composed of four star-shaped groups accessing four data sources, Dailymed, Diseasome, SIDER, and DrugBank. (b) Data Sources: Dailymed (RDF in Virtuoso), Diseasome (Local JSON File), SIDER (TSV in HDFS), DrugBank (XML in MySQL)

A snippet of sample raw data and relations is depicted in Figure 6.3. Our running query comprises eight (8) triple patterns (identified with $t1$ to $t8$ in the Figure 6.2a). Dailymed can answer the triple patterns $t1 - t4$, while triple pattern $t5$ can be answered by Diseasome. Further, SIDER can answer $t6 - t7$, and $t8$ can be answered by DrugBank. The data access services of each datasets are implemented by different backends and provide different capabilities. For instance, the endpoint services for SIDER and Diseasome are Spark-based query processors that translate queries from SPARQL to SQL, where the raw data need to be loaded in memory to evaluate the query in these data sources. Similarly, the endpoint for DrugBank translates SPARQL to SQL and execute the translated query in MySQL, which provides efficient indexing and query optimization for relational data.

Federated query engines, FedX [52] and MULDER [90], provide a unified view over a set of data sources that respect SPARQL protocol. They rely on source descriptions to select relevant sources for a given query and for finding an efficient query execution plan. For instance, FedX contacts the data sources to decide where the triple patterns will be executed, while MULDER requires RDF Molecule Temaplates (RDF-MTs) to be collected in advance. FedX decomposes the query, in Figure 6.2a, into five (5) sub-queries; $t1 - t3$, $t6 - t7$, and $t8$ that are sent to Dailymed, SIDER, and Drugbank, respectively, and $t4$ and $t5$ sent to all four (4) data sources, respectively. FedX creates a left linear tree plan with nested loop join operator, an operator that pushes down the join operation to the data sources by binding the join variables of the right operand with values extracted from the left operand, as shown in Figure 6.4a. FedX planner assumes the underlying data model is in RDF and triples are materialized in a triple store that is optimized for this data model. However, since the data sources have different data models and capabilities, pushing down join operations to the data sources would result in a higher execution time, 20min and incomplete results. On the other hand, MULDER decomposes the query into five (5) sub-queries; $t1 - t4$ executed in Dailymed, $t8$ executed in Drugbank, Diseasome executes $t5$, while $t6$ and $t7$ executed in SIDER, respectively. MULDER creates a bushy-tree plan with nested hash join and GJoin [34] operators based on the selectivity of operands to decide the type of operator. MULDER, like FedX, assumes RDF as an underlying data model and uniform querying capabilities of the given data sources. Based on these assumption, MULDER selects a nested hash join operator for the first two joins, between sub-queries ($t1 - t4$) vs ($t8$) vs ($t5$). Despite, creating an efficient bush-tree plan that helps to parallelize the query execution, the selection of join operator ignores the data source capabilities and underlying data model, which results in higher execution time, 4.6min. In this chapter, we devise optimization techniques guided by heuristics that enable the creation of source-dependent query plans. First, the Ontario query optimizer



```
dailymed:798    rdf:type                              dailymed:drugs ;
                dailymed:activeIngredient             dming:Simvastatin .
                owl:sameAs                            sider:54454 .
                dailymed:genericDrug                  drugbank:DB00641 ;
                dailymed:possibleDiseaseTarget        diseasome:319,
                                                      diseasome:2839,
                                                      diseasome:2175 .
```

Drug

accNum	DrugName	formula	pubChemId
DB00641	simvastatin	$C_{25}H_{38}O_5$	54454
DB00295	Morphine	$C_{17}H_{19}NO_3$	5288826

Drug_Target

Drug	Target
DB00641	631
DB00641	1882
DB00295	7683

Target

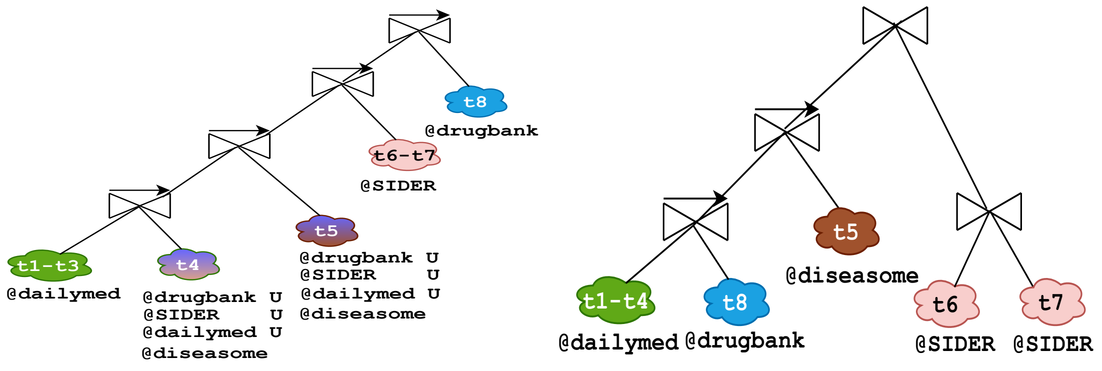
ID	Name	Gene	UniprotID
631	3-hydroxy-3-methylglutaryl-co enzyme A reductase	HMGCR	P04035
1882	Ras-related C3 botulinum toxin substrate 1	RAC1	P63000
7683	Mu-type opioid receptor	OPRM1	P35372

```
[{
  "diseaseID": "319",
  "name": "Diabetes_mellitus",
  "associatedGene": ["ACE", "ABCC8", "TCF1"]
  },{
  "diseaseID": "2839",
  "name": "Kaposi sarcoma, susceptibility to, 148000",
  "associatedGene": ["IL6", "IFNB2", "BSF2"]
 }]
```

```
side_effects.csv
DrugID,UMLS_ID,SideEffectName
54454,C0009806,Constipation
54454,C0236071,Throat tightness
54454,C0156404,Menstruation irregular
191,C0012833,Dizziness
191,C0232487, Abdominal discomfort
191,C1956346,Coronary artery disease

drug_names.csv
ID,DrugName
54454,simvastatin
191,adenosine
```

Figure 6.3: Example raw data in each data sources



(a) FedX Query Plan (b) MULDER Query Plan

Figure 6.4: **Motivating Example**. (a) FedX created a left-linear plan and used nested loop joins (arrows on top of join) (b) MULDER identifies a bushy-tree for star-shape groups.

resorts to data source descriptions in terms of RDF Molecule Templates to select the sources that will evaluate the query. Then, the query is decomposed into subqueries that can be executed in the selected sources. Finally, a plan that composes the subqueries is generated; physical operators are selected in order to minimize execution time and maximize answer completeness.

We tackle the problem of federated query processing over heterogeneous data sources in a Semantic Data Lake and propose ONTARIO, a query engine able to efficiently interoperate among heterogeneous datasets. ONTARIO implements novel query processing methods, i.e., source selection, query decomposition, and query planning; they are capable of exploiting knowledge about the sources and the query to generate plans *customized* for the sources in a Semantic Data Lake. ONTARIO resorts to RDF Molecule Templates in order to identify the star-shaped-group subqueries of an input query. Differently to state-of-the-art approaches, Ontario is able to classify star-shaped-group subqueries according to type of instantiations and joins. Additionally, star-shaped-group subqueries are characterized in terms of the data engines where they will be executed. Ontario exploits these meta-data to identify efficient query plans.

6.2 Problem Statement and Proposed Solution

Our formalization is based on RDF Molecule Templates, which represent an abstract description of entities stored in heterogeneous data sources that have the same semantic type.

Definition 15 (Semantic Data Lake) *A Semantic Data Lake (SDL) is a tuple $SDL = \langle \psi, \mathbb{S}, M \rangle$ where, ψ is a set of RDF Molecule Templates, $\mathbb{S}$ is a set of sources in raw formats (stored either in a file system or DBMS) in the Data Lake, M is a set of conjunctive rules that associate sources in $\mathbb{S}$ with RDF Molecule Templates in ψ.*

Definition 16 (Instantiation of an RDF-MT) *Instantiation of an RDF-MT, $[\sigma]$, is defined as a set of RDF molecules, σ^*, that are the instances of a class from data source(s) as described in the template.*

$$[\sigma] = \{\sigma^* | \forall p \in \sigma^*, p \subseteq \gamma, \ where\ \gamma \subseteq \sigma\} \tag{6.1}$$

Definition 17 (Virtual Knowledge Graph) *Given a Semantic Data Lake $SDL = \langle \psi = \{\sigma_1, \ldots, \sigma_k\}, \mathbb{S} = \{S_1, \ldots, S_n\}, M \rangle$, a Virtual Knowledge Graph (KG*) for SDL is a virtual RDF graph that corresponds to the union of all the RDF Molecules instantiations, σ^*, that are created by applying rules in M to the data sources in $\mathbb{S}$:*

$$KG^* = \bigcup_{i=1}^{n} \bigcup_{j=1}^{k} [\sigma_j]_{S_i} \tag{6.2}$$

In order to efficiently query the resulting Virtual Knowledge Graph, SPARQL queries need to be rewritten into queries operating on the data sources. SPARQL language is based on matching graph patterns; a basic graph pattern (BGP) is a set of triple patterns and (optional) filter clauses. A BGP in a SPARQL query contains at least one star-shaped subquery (SSQ), a non-empty set of triple patterns that share the same subject variable (constant). In order to efficiently query the resulting Virtual Knowledge Graph, SPARQL queries need to be rewritten into queries operating on the data sources.

Definition 18 (Query Rewriting) *Let Q and β(Q) be a SPARQL query and the set of Basic Graph Patterns (BGPs) in Q, respectively. Let $SDL = \langle \psi, \mathbb{S}, M \rangle$ be a Semantic Data Lake. A rewriting Q' of Q over sources in S corresponds to a SPARQL query, composed of BGPs in $\beta(Q')$ that meet the following conditions:*

- *β(Q) has the same number of triple patterns as β(Q'), i.e., $\tau(Q) = \tau(Q')$*
- *there is a function μ: β(Q) $\rightarrow$ β(Q') that maps BGPs in β(Q) to its corresponding rewriting in the sources of* SDL. $\mu\langle BGP_i \rangle = \{\langle BGP_{ij}, S \rangle | BGP_{ij} \subset BGP_i, S$ *is a non-empty set and* $S \subset \mathbb{S}\}$

6.2.1 Problem Statement

Given a SPARQL query Q, a Semantic Data Lake $SDL = \langle \psi, \mathbb{S}, M \rangle$, and a Virtual Knowledge Graph KG^* of *SDL*. The problem of federated query processing against heterogeneous data sources in a Semantic Data Lake (*SDL*) is defined as follows. Given a set of BGPs in Q, find a query Q' that satisfies the following conditions:

- The evaluation of Q over heterogeneous data source in *SDL* is complete, i.e., the evaluation of Q in KG^* is equivalent to the evaluation of Q' in *SDL*

$$[[Q']]_{SDL} = [[Q]]_{KG*} \tag{6.3}$$

- The cost of executing Q in *SDL* has a minimal execution cost, i.e., if $cost([[Q']]_{SDL})$ represents the execution time of Q' in *SDL*, then

$$[[Q]]_{SDL} = \underset{[[Q']]_{SDL}}{\text{argmin}} \ \text{cost}([[Q']]_{SDL}) \tag{6.4}$$

6.2.2 Proposed Solution

To tackle the problem of federated query processing against heterogeneous data sources in a Semantic Data Lake, we propose Ontario, a federated query processing engine over heterogeneous data sources in a Semantic Data Lake. Ontario utilizes the SPARQL query language as a unified query language and, its decomposition and source selection technique, similar to the MULDER approach (Chapter 5), is based on RDF Molecule Templates which describes set of entities which share similar semantic types (Chapter 4). The Ontario approach tackles the query planning and optimization sub-problem of federated query processing, aims to generate an execution plan that maximizes answer completeness and minimize the execution time. Given a SPARQL query, Ontario creates a set of star-shaped groups that matches RDF Molecule Templates in the Semantic Data Lake. Furthermore, Ontario is able to distinguish different types of star-shaped groups and decide which of them are more appropriate to be run in a given engine. The type of the star-shaped groups are further considered to decide the shape of the query plan tree, the more suitable join operators, and the location of the selections and projections in the plan. Ontario utilizes a greedy algorithm to find an efficient plan that minimized execution time and maximize answer completeness.

6.3 Ontario: Federated Query Processing over Semantic Data Lakes

Different subqueries, i.e., star-shaped subqueries, behave differently depending on the data source capabilities. Ontario is able to differentiate the following star-shaped groups:

1. CI: In this category, the star-shaped group do not have a constant object (instantiation) or a filter clause on object variables in any of triple patterns.
2. CII: Star-shaped groups in this category do not have a constant object or a filter clause in any of triple patters. Further, these star-shaped groups are defined over RDF-MTs described in terms of joins of two or more relations in the Data Lake.
3. CIII: Star-shaped groups in this category are composed of triple patterns with constant objects or contains filter clauses on object variables.
4. CIV: In addition to constant objects or filter clause, the star-shaped groups in this category are defined over RDF-MTs described in terms of joins of two or more relations in the Data Lake.

As to be shown in subsection 6.4.1, existing database engines (e.g., RDF or relational engines) may exhibit diverse performance during the execution of these star-shaped groups. For example, RDF engines will have expensive executions of star-shaped group in CI and CII, while relational engines will perform

Algorithm 4 Query Plan Generation: Φ - query decomposition, Q - SELECT query

```
1: procedure CreatePlan(Φ, Q)
2:     $\alpha \leftarrow []$
3:     for $SQ \in \Phi$ do
4:         orderTriples(SQ)
5:         $\alpha$.push(SQ)
6:     end for
7:     $P \leftarrow Q.projs()$                    ▷ Q.projs() - list of join and projected variables
8:     $\alpha \leftarrow$ OrderSSQs($\alpha, P$)
9:     while $len(\alpha) > 1$ do
10:        $SQ_i \leftarrow \alpha.pop()$
11:        $\delta \leftarrow []$
12:        $\beta \leftarrow [SQ_j \ for\ SQ_j \in \alpha$ if $shareVars(SQ_i, SQ_j)]$
13:        $\beta \leftarrow$ OrderSSQs($\beta, P$)
14:        for $SQ_j \in \beta$ do
15:            J $\leftarrow$ join($SQ_i$, $SQ_j$)
16:            $\alpha$.remove($SQ_j$)
17:            $\alpha$.push(J)
18:            break
19:        end for
20:        if $|\beta| = 0$ then
21:            $\delta$.push($SQ_i$)
22:        end if
23:    end while
24:    if $len(\delta) > 0$ then
25:        $\alpha \leftarrow$join($\alpha, \delta$)
26:    end if
27:    return $\alpha$
28: end procedure
```

badly on subqueries in CIII and CIV if no indexes exist over the instantiated or joined attributes. Contrary, since RDF engines always create indexes over the subject, predicates, and objects of the triples in an RDF graph [101], subqueries in CIII and CIV will be sped up in RDF engines.

6.3.1 Heuristics

Once the star-shaped groups and the sources where they will be executed are identified, the Ontario optimizer uses a set of heuristics to build query plans. These heuristics are guided by the general characteristics of the star-shaped groups, the type of star-shaped group, and the type of selected data sources.

General Heuristics

Ontario optimizer first consider the general characteristics of the star-shaped groups, referred to as general heuristics, while generating the execution plan of a given query. The general heuristics includes: percentage of constants in the star-shaped groups, ordering of triple patterns within a star-shaped group, number of projected variables within a star-shaped group, ordering of star-shaped groups based on selected data source types, and capability of storage engines for a selected data source.

Ordering between Star-Shaped Groups. Triple patterns with the highest number of constants at any part of the triple pattern are more selective. When ordering two star-shaped groups, a star-shaped group with the highest percentage of constants have higher precedence. Especially if bound join physical operators is selected, executing the more selective star-shaped group first could significantly reduces the size of intermediate results and minimize the execution time.

Triple Pattern Ordering within Star-Shaped Groups. The order of triple patterns in star-shaped group could affect the performance of the execution engine, e.g., triple stores. A set of triple patterns in a star-shaped group are ordered as follows: spo > sp > s > po > o > p. That is, a triple pattern with constants in all parts precedes a triple pattern with constants in subject and predicate. Similarly, a triple pattern with constants at subject and predicate precedes a triple pattern with a constant at subject part only, and so on.

Number of Projected Variables in Star-Shaped Groups. Given two star-shaped groups that have the same number of constant percentages, a star-shaped group with less number of projected variables have higher precedence. Star-shaped groups that project more values could consume more memory. This heuristics makes sure that such subqueries have a matching data with the rest of star-shaped groups to reduce memory usage.

Ordering based on Data Source Type. SSQs that are executed in endpoints with efficient indexes by a database management system (DBMS) are more selective than non-indexed raw file systems. For example, TripleStore >= RDBMS >= Graph > Document > HDFS > LOCAL_FILE.

Source Capabilities. A sub-query evaluated by a data source with capabilities to push down join and filter operation precedes a sub-query to be evaluated in a data source with less capabilities. Capabilities include, indexes, optimization, join operators, filters, etc. For example, MongoDB supports a poor join operation between documents.

Star-Shaped Group Based Heuristics

The second type of heuristics the Ontario optimizer uses is based on the type of star-shaped groups in a subquery. These heuristics are defined by both the category of star-shaped group and storage engine of the selected data source. They enable the transformation of subqueries in one category into subqueries in another category.

Pushing down instantiations into a Star-Shaped Group. This rule is performed whenever a star-shaped group SSQ_i of type CI is executed over an RDF engine in a query Q. If SSQ_i is part of a join in Q, SSQ_i is selected as the inner subquery of the join; a nested loop join is chosen as the physical operator. Additionally, if variables in triple patterns of SSQ_i are part of a filter in Q, the filter is represented as an instantiation of SSQ_i. Thus, this rule transforms subqueries in CI into subqueries in CIII.

Breaking up joins in Star-Shaped Groups This rule is performed whenever a star-shaped group SSQ_i of type CII is executed over an RDF engine in a query Q. In this case, SSQ_i is divided into as many subqueries as joins are defined in the corresponding RDF-MT and the attributes used in SSQ_i. These subqueries are connected by nested loop join operators that will be executed at Ontario level. Thus, this rule enables for transforming subqueries in CII into subqueries in CIV.

Pushing up instantiations into a Star-Shaped Group This rule is performed whenever a star-shaped group SSQ_i of type CIII is executed over an RDB engine in a query Q and the instantiation is not over an indexed attribute. If SSQ_i is part of a join in Q, hash join (or gjoin [34]) is chosen as the physical operator. Further, the selection is represented as a filter which is performed at Ontario level. Thus, this rule enables for transforming subqueries in CIII and CIV into subqueries in CI and CII, respectively.

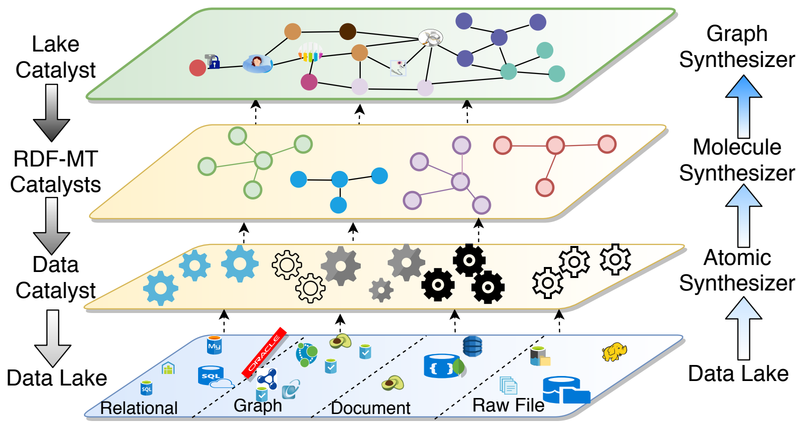


Figure 6.5: **Semantic Data Lake Basic Components:** Lake Catalyst catalyzes a SPARQL query into a set of star-shaped groups and select RDF-MTs matching each sub-query. RDF-MT Catalysts catalyze star-shaped groups into subqueries that can be executed in different Data Catalysts. Finally, Data Catalysts execute a sub-query in a data source by translating a given SPARQL query to native query language of the data source. Conversely, the results returned from each these components need to be synthesized (by Atomic, Molecule, and Graph Synthesizers) and passed to the upper layer.

Combining joins into a Star-Shaped Group This rule is performed whenever two star-shaped groups SSQ_i and SSQ_j of type CI are executed over an RDB engine, and there is a join between them over an indexed attribute. SSQ_i and SSQ_j are merged into one star-shaped group $SSQ_{i,j}$, transforming subqueries in CI into subqueries in CII.

6.3.2 Query Plan Generation

Given a SPARQL query, Ontario produces a decomposition composed of star-shaped groups (SSQs). Using RDF-MT descriptions, Ontario finds a matching RDF-MT for each SSQs. An RDF-MT matches an SSQ if it contains the same predicates as in SSQ. The selected SSQs composed a decomposition, Φ, which represents the input for the optimizer sketched in Algorithm 4. The optimizer is guided by the heuristics in section 6.3.1 and subsection 6.3.1. The planner first performs triple ordering within each SSQs, Line 3-5. The planner then orders SSQs in Φ based on their selectivity, categories, and the data sources, Line 7. Then, iteratively, it picks an SSQ_i (Line 9) and joinable SSQ_js (Line 11), i.e., that share same join variable and create join between them, Line 13-16. Joins are selected and SSQs are ordered according to the heuristics in subsection 6.3.1. Figure 6.6 illustrates the Ontario architecture.

6.3.3 The Ontario Architecture

In this section, we present the architecture of Ontario. First, we present an overview of basic components in a Semantic Data Lake architecture, then we describe the Ontario architecture with respect to these components. Figure 6.5 shows an overview of the Semantic Data Lake components. The top layer catalyzes, i.e., via the Lake Catalyst, a given SPARQL query into a set of star-shaped groups and selects matching RDF Molecule Templates. The Lake Catalyst passes the sub-queries to the respective RDF-MT Catalysts that are responsible for specific RDF Molecule Templates. It dispatches the decomposed query and coordinates the global query planing and optimization. In addition, once the results are returned from lower layer, i.e., from Molecule Synthesizers, the Graph Synthesizer combines (synthesizes) the molecules and generate the final result. The second layer catalyzes, i.e., via RDF-MT Catalysts, the star-shaped groups into a set of API calls to the Data Catalysts, which then transform the raw data

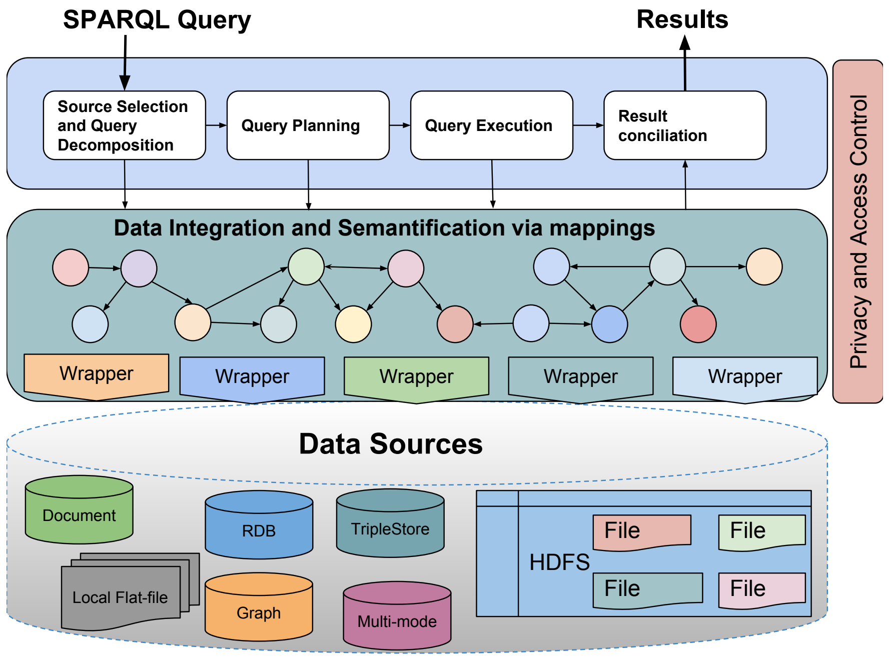


Figure 6.6: The Ontario Architecture

on-the-fly and synthesize results, i.e., via Molecule Synthesizer, returned from Atomic Synthesizers. RDF-MT Catalysts are specialized components that deal with only one specific RDF Molecule Template, and provide decomposition, planning and execution of a particular star-shaped group. The third layer provide access to a specific data source by translating queries from global querying mechanism, e.g., SPARQL, to the underlying native query mechanism of the data sources via Data Catalysts. Atomic Synthesizers, on the other hand, transform raw data to RDF on demand by applying mapping rules, e.g., defined by RML or R2RML mapping languages. They are specialized to a specific data model and system interface. Atomic Synthesizers perform transformation of the results from native data sources to RDF based on transformation rules. The bottom layer, Data Lake, provides an infrastructure to store raw data and access interface to a set of heterogeneous data sources. These data sources can be characterized with different properties, such as autonomy (sources can be autonomous), data format heterogeneity (provides different data formats), access interface heterogeneity (various query languages), semantic heterogeneity (different representation of same data points), volume (different sizes from small to large data sets), access restrictions, etc.

In Figure 6.6, the Ontario architecture is presented. Given a SPARQL query, Ontario decomposes into a set of star-shaped groups with matching relevant RDF Molecule Templates. For each sub-query the set of RDF-MT Catalysts that can evaluate each matching RDF-MT are selected. Given a set of star-shaped groups and relevant data sources for matching RDF-MTs, Ontario query planner creates a bushy-tree plan utilizing the heuristics described in Section 6.3.1. The query optimizer, optimizes the generated plan and selects the appropriate operators based on the source capability and source type heuristics. The corresponding RDF-MT Catalysts then execute each sub-query, which internally contacts the appropriate Data Catalysts to synthesize the (sub-graph of) RDF Molecule Templates. Finally, the Ontario Lake Catalyst merges the results to synthesize the final results and return back to the user. During the catalysis of the given SPARQL query in the Lake Catalyst, RDF-MT Catalysts as well as Data Catalysts, the privacy and access rules set by each data sources must be respected. Privacy and access control constrains must be checked as well during synthesis of results by Atomic Synthesizer, Molecule Synthesizer, and

Data Model	Data Sources
RDF	KEGG, ChEBI, Diseasome, LinkedCT, Dailymed
RDB	DrugBank, SIDER, Medicare, Affymetrix, TCGA

Table 6.1: **Distribution of Data Sources over Data Formats**

Graph Synthesizer.

6.4 Empirical Evaluation

We empirically study the behavior of Ontario; it is compared with the state-of-the-art RDF federated engines FedX [52] and MULDER [90]. We study the following research questions: **Q1)** What is the overhead of considering heterogeneity during federated query processing? **Q2)** Can RDF-MT based source descriptions be effectively applied for source selection, query decomposition, and optimization for non-RDF data sources? **Q3)** Are Ontario optimization techniques able to generate effective and efficient query plans for heterogeneous data sources? The experimental configuration to evaluate these research questions is as follows:
Benchmark: LSLOD [93] is a benchmark composed of ten real-world datasets of the Linked Open Data (LOD) cloud from the life sciences domain. The RDF version of LSLOD datasets are transformed into RDB tables. Initially, all the RDF triples that correspond to an RDF-MT are included in one table, but functional and multivalue dependencies between the attributes of a table are utilized to produce normalized version of the table in 3NF. Thus, attributes containing multiple values for one subject are stored in a separate table. Tables and RDF graphs of each of the LSLOD datasets are uploaded in a dedicated Docker container. RDB tables are loaded into *MySQL 5.7.24* and indexes are created for the primary key of each table. We study the LSLOD simple queries [93].
Metrics: We report on the following metrics: **a)** *Execution Time*: Elapsed time between the submission of a query to an engine and the delivery of the answers. Time corresponds to absolute wall-clock system time as reported by the Python `time.time()` function. Timeout is set to 300 seconds. **b)** *Cardinality*: Number of answers returned by a query. **c)** *Completeness*: Query result percentage with respect to the answers produced by the unified SPARQL endpoint created as the union of all datasets in the benchmark. **d)** ***dief@t***: measures the continuous efficiency of an engine in the first *t* time units of query execution.
Implementation: Ontario [5] is implemented in Python 3.6. Currently, Ontario employs wrappers for MySQL, PostgreSQL, MongoDB, Neo4j, and wrappers for flat files (TSV, CSV, JSON, and XML) stored either in HDFS or local file system. Two versions of Ontario are compared: **i) RDF version** for federated queries against RDF graphs accessible via SPARQL endpoints; and **ii) RDF+RDB version** for federated queries against RDF graphs accessible via both SPARQL endpoints and RDB tables stored in MySQL.

6.4.1 Impact of Star-shaped Group Types

In this experiment, we analyze the impact of different star-shaped groups (SSQs) on the performance of the query engine. This analysis allows us to understand the behaviors of the engines while evaluating sub-queries and adopt in the planning and execution strategy. Figure 6.7 shows the performance of Ontario Semantic Data Lake query engine while performing semantification on-demand, compared to execution over materialized version of the same dataset in RDF for selected star-shaped groups in each category. The behaviour of the engine on SSQs in categories CI ($SSQ_{1,1}$, $SSQ_{1,2}$, and $SSQ_{10,2}$) and CII (i.e., $SSQ_{6,1}$, and $SSQ_{8,1}$), is presented in Figure 6.7a. As can be observed, star-shaped groups in these

[5] `https://github.com/SDM-TIB/Ontario`

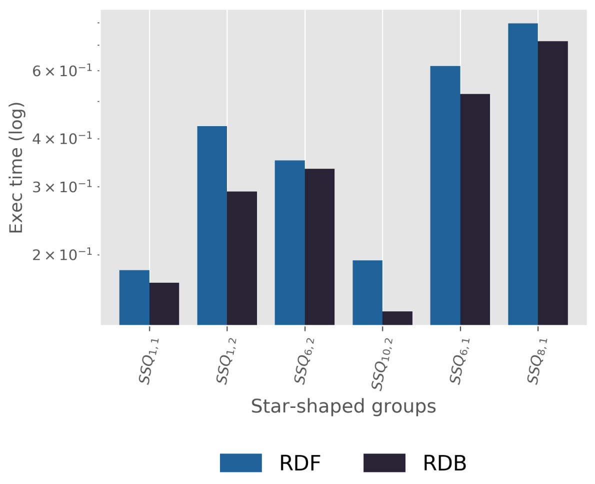
Query	**Sub-query**	Category	Source Type	Source(s)
SQ_1	$SSQ_{1,1}$	CI	RDB	DrugBank
	$SSQ_{1,2}$	CI	RDF	Dailymed
SQ_2	$SSQ_{2,1}$	CI	RDF	KEGG
	$SSQ_{2,2}$	CII	RDF	KEGG
	$SSQ_{2,3}$	CIV	RDB	DrugBank
SQ_3	$SSQ_{3,1}$	CI	RDB	DrugBank
	$SSQ_{3,2}$	CI	RDF	ChEBI
	$SSQ_{3,3}$	CIV	RDF	KEGG
SQ_4	$SSQ_{4,1}$	CIV	RDB	DrugBank
	$SSQ_{4,2}$	CIII	RDF	KEGG
SQ_5	$SSQ_{5,1}$	CIV	RDB	DrugBank
	$SSQ_{5,2}$	CIV	RDF	KEGG $\cup$ ChEBI
	$SSQ_{5,3}$	CI	RDB	DrugBank
SQ_6	$SSQ_{6,1}$	CII	RDB	DrugBank
	$SSQ_{6,2}$	CI	RDF	Diseasome
SQ_7	$SSQ_{7,1}$	CIV	RDF	Dailymed
	$SSQ_{7,2}$	CI	RDB	SIDER
	$SSQ_{7,3}$	CII	RDB	SIDER
SQ_8	$SSQ_{8,1}$	CII	RDB	DrugBank
	$SSQ_{8,2}$	CI	RDF	Diseasome
SQ_9	$SSQ_{9,1}$	CIV	RDF	LinkedCT
	$SSQ_{9,2}$	CII	RDF	Dailymed
SQ_{10}	$SSQ_{10,1}$	CIV	RDF	LinkedCT
	$SSQ_{10,2}$	CI	RDB	DrugBank

Table 6.2: **Characteristics of SSQs of the LSLOD benchmark Queries**. LSLOD queries are described in terms of categories of star-shaped-groups (SSGs). Categories are as follows: CI with no instantiations of properties; CII with no instantiations and joins at the RDF-MT definition; CIII with instantiations, and no joins at the RDF-MT definition; and CIV with instantiations with joins at the RDF-MT definition.

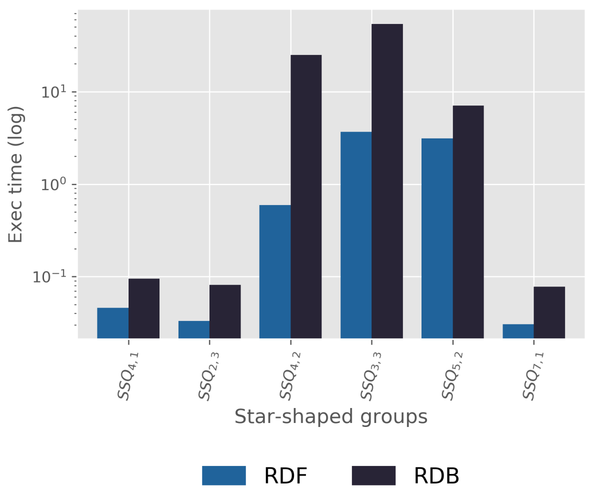
categories are more expensive in RDF than RDB. RDF engines have indexes over combination of subject, predicate, and object. When a triple pattern do not have instantiation ether in subject or object part of any triple patterns, then the engine scan all available data for each predicate in the subquery. On the other hand, relational engine create indexes on primary keys (and optionally any other columns). In RDB even if triple patterns do not have instantiations, they only scan a relation or a set of relation, unlike RDF triple stores that scan over all data. This leads to RDB engines perform better in these categories than RDF engines. Figure 6.7b presents SSQs in category CIII and CIV, where there are triple patterns with object instantiations. The behaviour of the engines in this category shows that, RDF engine performs faster than RDB. This entails, RDB engine performs slower than RDF when the instatiations are not on the indexed predicates, and allows for answering **Q1**.

6.4.2 Impact of Considering Heterogeneity

In this experiment, we evaluate the performance of Ontario in federation of RDF data sources and an overhead introduced while considering heterogeneity during query execution compared to FedX and MULDER. Figure 6.8 presents the results of executing LSLOD queries over RDF data source. Figure 6.8a

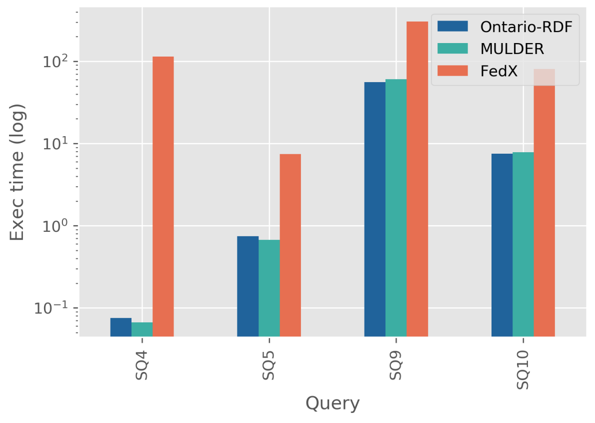


(a) Category I and II

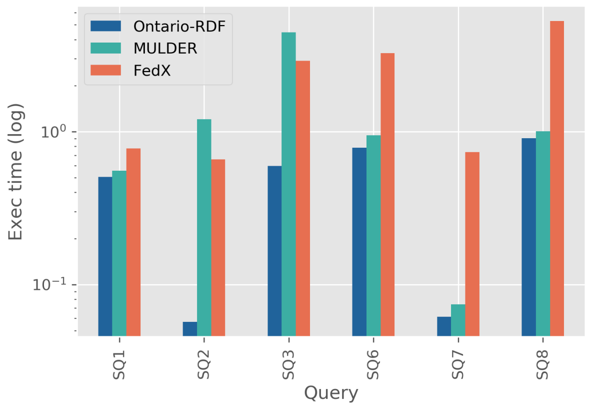


(b) Category III and IV

Figure 6.7: **Star-Shaped group types**. The impact of Star-Shaped group types is reported.



(a) Same plan by ONTARIO and MULDER



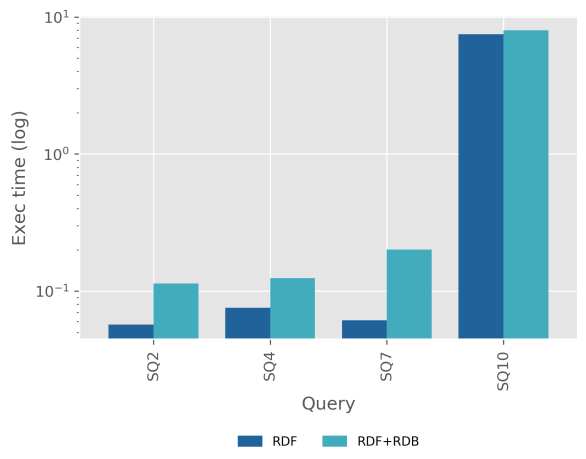
(b) Queries with improved performance

Figure 6.8: **Efficiency of ONTARIO on homogeneous data sources.** ONTARIO is compared with existing SPARQL federation engines, MULDER and FedX. (a) ONTARIO outperform FedX in all queries. (b) ONTARIO overcomes both FedX and MULDER by generating efficient plans and using optimization rules tailored for RDF sources.

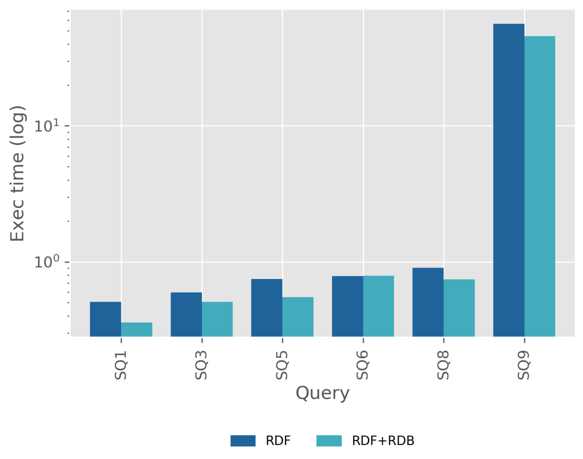
suggests that ONTARIO outperforms FedX on queries $SQ4$, $SQ5$, $SQ9$, and $SQ10$. Even though, ONTARIO generates same plan as MULDER for these queries, it pays a price for considering heterogeneous data sources compared to MULDER for $SQ4$ and $SQ5$ queries. On the other hand, ONTARIO outperform both FedX and MULDER by generating efficient plans and using optimization rules tailored for RDF sources on the rest of the queries, as shown in Figure 6.8b. These results also allow for answering **Q1**.

6.4.3 Impact of Heterogeneous Sources

The performance of ONTARIO over heterogeneous source, i.e., RDF and RDB, is evaluated and analyzed with respect to the SSQ categories. Figure 6.9 presents the results of executing LSLOD queries over two versions of ONTARIO: RDF only and RDF+RDB. For queries that are composed of SSQs in CIII and CIV, i.e., sub-queries with object intantiations, the RDF version performs better than RDF+RDB, as



(a) Queries composed of SSQs in CIII or CIV



(b) Queries composed of SSQs in CI or CII

Figure 6.9: **Performance of Ontario engine on heterogeneous sources.** Executing queries composed of SSQs in Category III or IV are expensive in RDF+RDB data sources, whereas SSQs in Category I and II are expensive in RDF only data sources.

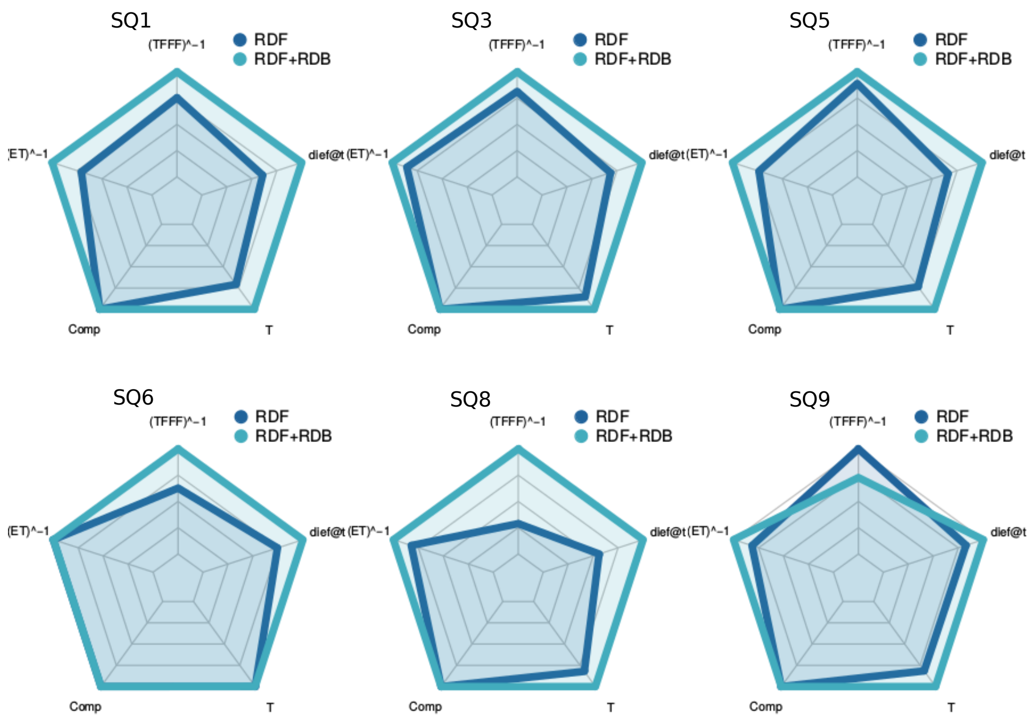


Figure 6.10: **Queries where RDF+RDB is better**. $(TFFF)^{-1}$ - inverse time for first result, ET^{-1} - inverse execution time, Comp - Completeness, T - throughput, and $dief@t$ continuous efficiency in time t

shown in Figure 6.9a. This is expected, as we have observed in the Experiment I, star-shaped groups with instatiations are cheaper in RDF than RDB. On the other hand, for queries that are composed of SSQs in CI and CII, the RDF+RDB version performs faster than the RDF version, as shown in Figure 6.9b. These results also allow for answering **Q2** and **Q3**.

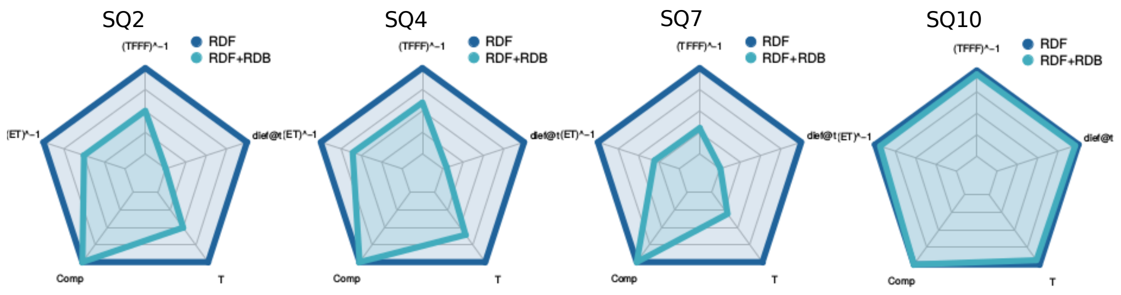


Figure 6.11: **Queries where RDF only is better**. $(TFFF)^{-1}$ - inverse time for first result, ET^{-1} - inverse execution time, Comp - Completeness, T - throughput, and $dief@t$ continuous efficiency in time t

6.4.4 Measuring the Continuous Efficiency

Figure 6.10 and Figure 6.11 report on the performance of Ontario in producing continuous answers. The continuous efficiency in time t, i.e., $dief@t$, Inverse of Time for the first tuple ($TFFF^{-}1$), Inverse of Total Execution ($TE^{-}1$), Number of answers produced (Comp), and Throughput (T), are presented in Figure 6.10 and Figure 6.11 using radar plots. The interpretation of these metric in each axes is **'higher is better'**. For all queries, the completeness (Comp) of the queries is 100%, but the throughput varies as it correlates with the overall execution time. As clearly shown, the continuous efficiency of the RDF+RDB version is better in $SQ1$, $SQ3$, $SQ5$, $SQ6$, $SQ8$, and $SQ9$, while it is lower in $SQ2$, $SQ4$, and $SQ7$ than the RDF version. These results are aligned with the previous experiments and answer **Q2** and **Q3**.

6.5 Summary

We presented Ontario, a federated query processing engine over heterogeneous data sources in a Data Lake. Ontario relies on RDF Molecule Templates to describe heterogeneity of data sources; it is also able to decompose a SPARQL query into a set of star-shaped groups that can be efficiently executed. Ontario also identifies bushy-tree plans which are able to reduce execution time and increase answer completeness. The Ontario optimizer is guided by a set of heuristics defined at the level of star-shaped groups and the data engines where they will be executed. We showed through our empirical analysis that, even though, data engines behave differently on diverse types of star-shaped groups, Ontario is able to create efficient and effective plans where physical operators are selected accurately. Thus, our work expands the series of techniques available for federated query processing, and we hope that our techniques will provide scalable solutions in real-world settings.

CHAPTER 7

Privacy-aware Query Processing

In recent years, the amount of both open data available on the Web and private data exchanged across companies and organizations has been constantly increasing. To address this new challenge of effective and efficient data-centric applications built on top of this data, i.e., data management techniques targeting sensitive data such as financial transactions, medical procedures, or various other personal data must consider various privacy and access control regulations and enforce privacy constraints once data is being accessed by data consumers. Existing works suggest the specification of access control ontologies for RDF data [71, 73] and their enforcement on centralized or distributed RDF stores (e.g., [75]) or federated RDF sources (e.g., [72]). Albeit expressive, these approaches are not able to consider privacy-aware regulations during the whole pipeline of a federated query processing engine; during source selection, query decomposition, planning, and execution. As a consequence, efficient query plans cannot be devised in a way that privacy and access policies are enforced. Previous chapters tackles the problem of federated query processing at the level of query decomposition and source selection,Chapter 5, as well as the level of query planning and optimization, Chapter 6, sub-problems. However, these techniques assumed the data is open without any restrictions. In this chapter, we tackle the problem of enforcing privacy and access control regulations during query processing and propose BOUNCER, a privacy-aware query engine over federations of data sources. The content of this chapter is based on the publication [102]. The result of this chapter provides an answer to the following research question:

RQ3: How can privacy and access control requirements be encoded in data source descriptions and be used for enforcement during query processing over federation of data sources?

To answer this research question, we propose a privacy-aware RDF Molecule Template based source description that define the type of operations allowed at the different level of granularity. Furthermore, we propose a privacy-aware federated query processing engine, BOUNCER, that is able to select relevant data sources respecting the privacy and access policies defined by the data source providers described in RDF-MTs and generate a valid execution plan that minimize execution time and maximize answer completeness without revealing sensitive information restricted by the privacy and access policies. Figure 7.1 shows the challenge tackled in this chapter and the contribution of in this chapter.

The contributions of this chapter can be summarized as follows:

- A thorough formalization of privacy-aware query processing problem over federation of autonomous data sources.

- A privacy-aware query decomposition and planning algorithms that generate a valid plan according to privacy and access control policy of data sources in the federation.

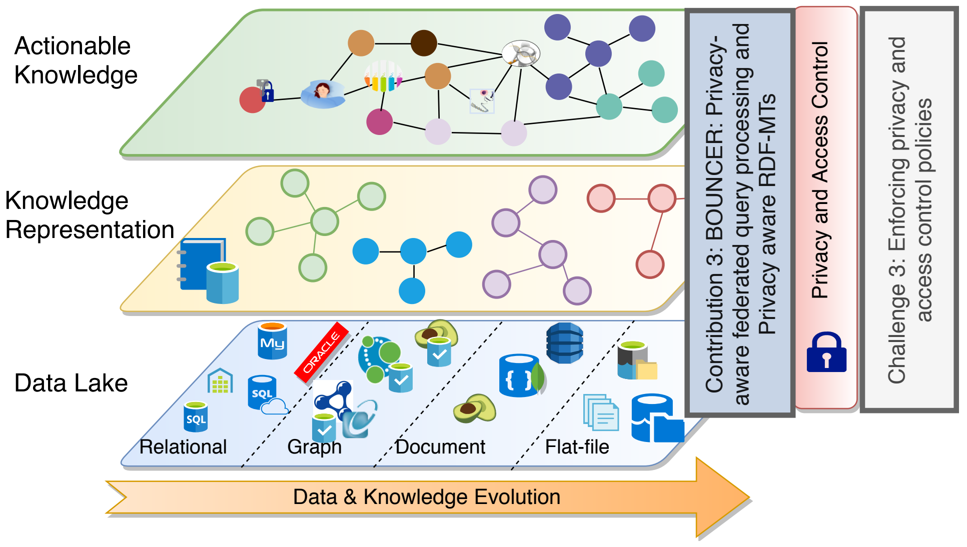

Actionable Knowledge
Knowledge Representation
Data Lake
Relational
Graph
Document
Flat-file
Data & Knowledge Evolution
Contribution 3: BOUNCER: Privacy-aware federated query processing and Privacy aware RDF-MTs
Privacy and Access Control
Challenge 3: Enforcing privacy and access control policies

Figure 7.1: **Challenges and Contributions**.This chapter focuses on the problem of privacy and access control federated query processing.

- An empirical evaluation assessing the effectiveness and efficiency of the proposed privacy-aware query processing technique. The observed results suggest that BOUNCER can effectively enforce access control regulations at different granularity without impacting the performance of query processing.

This chapter is structured as follows: Section 7.1 we motivate the problem of privacy-aware query processing over a federation of data sources using a real case scenario from medical domain. In Section 7.2, we formally define the problem of privacy-aware query decomposition and planning over a federation of data sources. Then, we present BOUNCER privacy-aware query processing engine in Section 7.3. Section 7.4 presents the results of an empirical evaluation of the proposed privacy-aware query processing techniques. Finally, Section 7.5 presents the closing marks of this chapter.

7.1 Motivating Example

We motivate our work using a real-world use case from the biomedical domain where data sources from clinical records and genomics data have been integrated into an RDF graph. For instance, Figure7.2

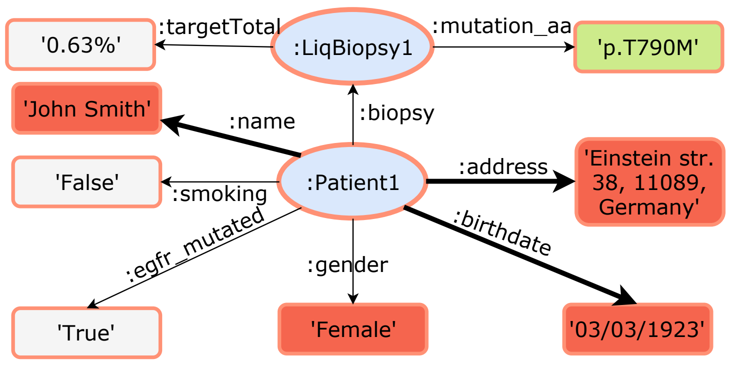


Figure 7.2: **Hospital Data (S1).** An RDF molecule representing a lung cancer patient; thicker arrows correspond to controlled properties.

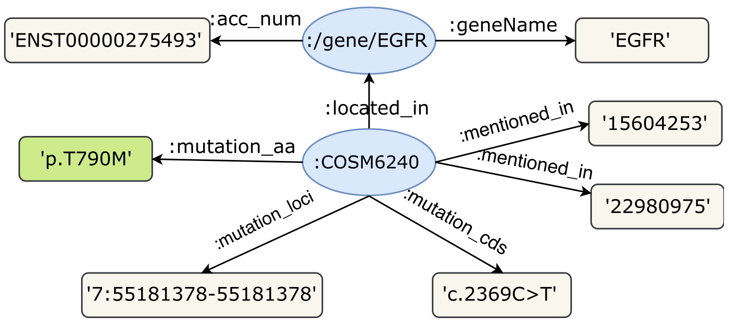
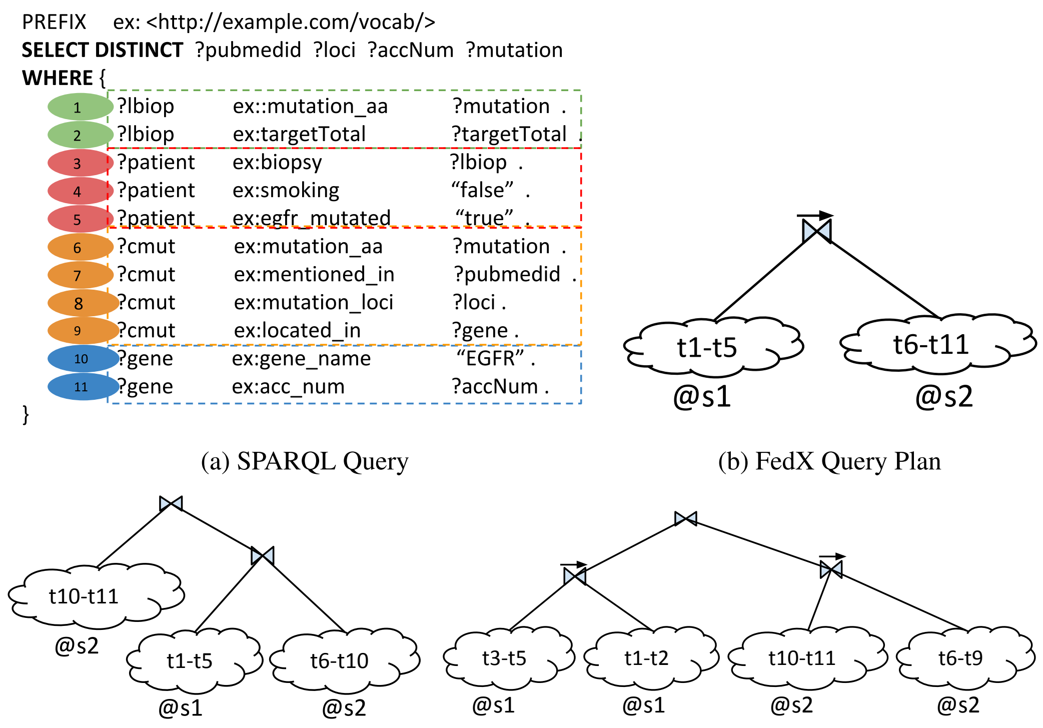


Figure 7.3: **Research Institute Data (S2).**An RDF molecule representing the results of a liquid biopsy of a patient. Servers at the hospital can perform join operations.

and Figure7.3 depicts two RDF subgraphs or RDF molecules. One RDF molecule represents a patient and his/her clinical information provided by source (S1), Figure7.2, while the other RDF molecule models the results of liquid biopsy available in a research institute (S2), Figure7.3. The privacy policy enforced at the hospital data source states that *projection (view)* of values is not permitted. Properties name, date of birth, and address of a patient (thicker arrows in Figure7.2) are controlled, i.e., query operations are not permitted. Furthermore, it permits a *local join operation* (on premises of the hospital data server) of properties, such as `ex:mutation_aa` - peptide sequence changes that are studied for a patient, `ex:targetTotal` - percentage of circulating tumor DNA in the blood sample of liquid biopsy, `ex:egfr_mutated` - whether the patient has mutations that lead to EGFR over-expression, and `ex:smoking` - whether the patient is a smoker or not. Suppose a user requires to collect the Pubmed ID, mutation name, the genomic coordinates of the mutation and accession numbers of the genes associated with non-smoking lung cancer patients whose liquid biopsy has been studied for somatic mutations that involve EGFR gene amplification (over-expression). Figure7.4a depicts a SPARQL query that represents this request; it is composed of 11 triple patterns. The first five triple patterns are executed against S1 while the last six triple patterns are evaluated over S2.

Existing federated query engines are able to generate query plans over these data sources. Figure7.4b shows a query execution plan generated by FedX [52] federated query engine for the given query. FedX decomposes the query into two subqueries that are sent to each data source. FedX uses a nested loop join operator to join results from both sources. This operator pushes down the join operation to the data sources by binding the join variables of the right operand of the operator with values extracted from the left operand. First, triple patterns from $t1 - t5$ are executed on S1, extracting values for the variables `?mutation_aa`, `?lbiop`, `?targetTotal`, and `?patient`. Then, the shared variable, `?mutation_aa`, is bound and the triple patterns $t6 - t11$ are executed over S2. However, executing this plan yields no answer since the privacy-policy of the hospital does not allow projection of values from the first subquery. Figure7.4c shows the query execution plan generated by ANAPSID [34] federated query engine. ANAPSID creates a bushy plan where join operation is performed using GJoin operator (special type of symmetric hash join operator). This operator executes the left and right operands and makes join on the federated engine. In order to check whether the results returned from the subqueries on the left and right operand can be joined, the values of shared variables from both operands have to be checked by ANAPSID, which requires extracting all values for all variables in both sources. This ignores the privacy policy enforced which yields no answer for the given query. The MULDER [89] federated query engine generates a bushy plan and decomposes the query by identifying matching RDF Molecule Templates (PRDF-MTs) as a subquery, as shown in Figure7.4d. PRDF-MT is a template that represents a set of RDF molecules that share the same RDF type (rdf:type). MULDER assigns nested hash join operator to join



```
PREFIX    ex: <http://example.com/vocab/>
SELECT DISTINCT  ?pubmedid  ?loci  ?accNum  ?mutation
WHERE {
   1  ?lbiop       ex::mutation_aa        ?mutation  .
   2  ?lbiop       ex:targetTotal         ?targetTotal .
   3  ?patient     ex:biopsy              ?lbiop  .
   4  ?patient     ex:smoking             "false"  .
   5  ?patient     ex:egfr_mutated        "true"  .
   6  ?cmut        ex:mutation_aa         ?mutation  .
   7  ?cmut        ex:mentioned_in        ?pubmedid  .
   8  ?cmut        ex:mutation_loci       ?loci .
   9  ?cmut        ex:located_in          ?gene .
  10  ?gene        ex:gene_name           "EGFR" .
  11  ?gene        ex:acc_num             ?accNum .
}
```

(a) SPARQL Query (b) FedX Query Plan

(c) ANAPSID Query Plan (d) MULDER Query Plan

Figure 7.4: **Motivating Example**. (a) A SPARQL query composed of four star-shaped subqueries accessing controlled and public data from S1 and S2. (b) FedX generates a plan with two subqueries. (c) ANAPSID decomposed the query into three subqueries. (d) MULDER identifies a plan with four star-shape subqueries. None of the query plan respects privacy policies of S1 and S2.

triple patterns $t3 - t5$ associated with `Patient` PRDF-MT and triple patterns $t1 - t2$ that are associated with `Liquid_Biopsy` PRDF-MT. Like in FedX, this operator extracts values for join and projection variables from the left operand, and then binds them to the same variables of the right operand. Like FedX and ANAPSID plans, the MULDER plan also ignores the privacy policy enforced at the hospital data source, which would yield an empty query answer. All of these federated engines fail to answer the query, because they ignore the privacy policy of the data sources during query decomposition as well as query execution plan generation (e.g., wrong join ordering). Also, MULDER ignores the privacy policy of the hospital during query decomposition and splits the triple patterns from this source. This leads to trying to extract results on the federation system which is not possible because of the restrictions enforced by the hospital. In addition to the join order problem, ANAPSID selects a wrong join operator which requires data from S1 to be projected for the restricted properties, i.e., $t1 - t5$. In this chapter, we present BOUNCER a privacy-aware federated query engine able to identify plans that respect the above-mentioned privacy and access control policies.

7.2 Problem Statement and Proposed Solution

In this section, we formalize the problem of privacy-aware query decomposition over a federation of RDF data sources. First we define a set of privacy-aware predicates that represent the type of operations that can be performed over an RDF dataset according to the access regulations of the federation.

Definition 19 (Privacy-Aware Operations) *Given a federated query engine $\mathcal{M}$, a federation $\mathcal{F}$ of RDF datasets D, and a dataset D_i in D. Let p_{ij} be an RDF property with domain the RDF class C_{ij}. The set of operations to be executed by $\mathcal{M}$ against $\mathcal{F}$ is defined as follows:*

- *join_local(D_i, p_{ij}, C_{ij}) - this predicate indicates that the join operation on property p_{ij} can be performed on the dataset D_i.*
- *join_fed(D_i, p_{ij}, C_{ij}) - this predicate indicates that the join operation on property p_{ij} can be performed by $\mathcal{M}$. The truth value of join_fed(D_i, p_{ij}, C_{ij}) implies to the truth value of join_local(D_i, p_{ij}, C_{ij}).*
- *project(D_i, p_{ij}, C_{ij}) - this predicate indicates that the values of the property p_{ij} can be projected from dataset D_i. The truth value of project(D_i, p_{ij}, C_{ij}) implies to the truth value of join_fed(D_i, p_{ij}, C_{ij}).*

Definition 20 (Access Control Theory) *Given a federated query engine $\mathcal{M}$, a set of RDF datasets $D = \{D_1, .., D_n\}$ of a federation $\mathcal{F}$. An Access Control Theory is defined as the set of privacy-aware operations that can be performed on property p_{ij} of RDF class C_{ij} over dataset D_i in D.*

The access control theory for the federation described in our running example of Figure7.4a can be defined as a conjunction of the following operations:

- *join_local(s1, ex:mutation_aa, Liquid_Biopsy),*
- *join_local(s1, ex:biopsy, Patient), project(s2, ex:located_in, Mutation),*
- *join_local(s1, ex:targetTotal, Liquid_Biopsy), project(s2, ex:acc_num, Gene),*
- *join_local(s1, ex:smoking, Patient), join_local(s1, ex:egfr_mutated, Patient),*
- *project(s2, ex:mutation_aa, Mutation),project(s2, ex:gene_name, Gene),*
- *project(s2,ex:mutation_loci,Mutation),project(s2,ex:mentioned_in,Mutation).*

Note that the RDF properties `:name`, `:gender`, `:address`, and `:birthdate` of the `Patient` RDF class do not have operations defined in the access control theory. In our approach this fact indicates that these properties are controlled and any operation on these properties performed by the federated engine is forbidden.

Definition 21 *Given a property p_{ij} of an RDF class C_i from a dataset D_i in a federation $\mathcal{F}$ and an access control theory T. If there is no privacy-aware predicate in T that includes p_{ij}, then p_{ij} is a* controlled property *and no federation engine can perform operations over p_{ij} against D_i.*

A basic graph pattern (BGP) in a SPARQL query is defined as a set of triple patterns $\{t_1, \ldots, t_n\}$. A BGP contains one or more triple patterns that involve a variable being projected from the original SELECT query. We call these triple patterns *projected triple patterns*, denoted as $PTP = \{t_1, \ldots, t_m\}$ such that $PTP \subseteq BGP$. A BGP includes at least one star-shaped subquery (SSQ), i.e., $BGP = \{SSQ_1, \ldots, SSQ_n\}$. A star-shaped subquery is a set of triple patterns that share the same subject variable or object [103]. Furthermore, an SSQ may contain zero or more triple patterns that involve a variable which is being projected from the original SELECT query. We call these triple patterns *projected triple patterns of an SSQ*, denoted as $PTS = \{t_1, \ldots, t_k\}$ where $PTS_i \subseteq SSQ_i$. Let *PRJ* be a set of triple patterns that involve a variable being projected from the original SELECT query, then projected triple patterns of a BGP, is a subset of PRJ, i.e., $PTP \subseteq PRJ$ and a projected triple pattern of SSQ_i is a subset of PTP, i.e., $PTS_i \subseteq PTP$. For example, in our running example, there is only one BGP, $BGP_1 = \{t_1, \dots, t_{11}\}$, for which projected variables belong to triple patterns, $PRJ = \{t_6, t_7, t_8, t_{11}\}$. Projected triple patterns of BGP_1 are the same as PRJ, $PTP_{BGP_1} = \{t_6, t_7, t_8, t_{11}\}$, since there is only one BGP. Furthermore, BGP_1 can be clustered into four start-shaped subqueries, $SSQs_{BGP_1} = \{SSQ_{1=\{t_1-t_2\}}, SSQ_{2=\{t_3-t_5\}}, SSQ_{3=\{t_6-t_9\}}, SSQ_{4=\{t_{10}-t_{11}\}}\}$. Out of four $SSQs$ of BGP_1, only the last two $SSQs$ have triple patterns that are also in the projected triple patterns, i.e., $PTS_{SSQ_1} = \emptyset$, $PTS_{SSQ_2} = \emptyset$,$PTS_{SSQ_3} = \{t_6, t_7, t_8\}$, $PTS_{SSQ_4} = \{t_{11}\}$.

Definition 22 *Given a SPARQL query Q such that a variable ?v is associated with a property p of a triple pattern t in a BGP and ?v is projected in Q. Suppose an access control theory T regulates the access of the datasets in D of the federation $\mathcal{F}$. A federation engine $\mathcal{M}$ accepts Q iff there is a privacy-aware operation $project(D_i, p, C)$ in T for at least an RDF dataset D_i in D.*

A privacy-aware query decomposition on a federation is defined. This formalization states the conditions to be met by a decomposition in order to be evaluated over a federation by enforcing their access regulations.

Definition 23 (Privacy-Aware Query Decomposition) *Let BGP be a basic graph pattern, PTP a set of projected triple patterns of a BGP, T an access control theory, and $D = \{D_1, \dots, D_n\}$ a set of RDF datasets of a federation $\mathcal{F}$. A privacy-aware decomposition P of BGP in D, $\gamma(P|BGP, D, T, PTP)$, is a set of decomposition elements, $\Phi = \{\phi_1, .., \phi_k\}$, such that ϕ_i is a four-tuple, $\phi_i = (SQ_i, SD_i, PS_i, PTS_i)$, where:*

- *SQ_i is a subset of triple patterns in BGP, i.e., $SQ_i \subseteq BGP$, and $SQ_i \neq \emptyset$, such that there is no repetition of triple patterns, i.e., If $t_a \in SQ_i$, then $!\exists t_a \in SQ_j : SQ_j \subset BGP \wedge i \neq j$,*
- *SD_i is a subset of datasets in D, i.e, $SD_i \subseteq D$, and $SD_i \neq \emptyset$,*
- *PS_i is a set of privacy-aware operations that are permitted on triple patterns in SQ_i to be performed on datasets in SD_i and $PS_i \subseteq T$, and $PS_i \neq \emptyset$,*
- *PTS_i is a set of triple patterns in SQ_i that contains variables being projected from the original SELECT query, i.e., $PTS_i \subseteq SQ_i \wedge PTS_i \subseteq PTP$,*
- *The set composed of SQ_i in the decompositions $\phi_i \in \Phi$ corresponds to a partition of BGP and*
- *The selected RDF datasets are able to project out the attributes in the project clause of the query, i.e., $\forall t_a \in SQ_i : t_a \in PTP$, then $project(D_a, p_{aj}, C_{aj}) \in PS_i$ where $t_a = (s, p_{aj}, o)$, $D_a \in SD_i$, and $SQ_i \in \phi_i$.*

After defining what is a decomposition of a query, we state the problem of finding a suitable decomposition for a query and a given set of data sources.

Privacy-Aware Query Decomposition Problem. Given a SPARQL query Q, RDF datasets D=$\{$D$_1$,...,D$_m\}$ of a federation $\mathcal{F}$, and access control theory T. The *problem of decomposing Q* in D restricted by T is defined as follows. For all BGPs, BGP=$\{$t$_1$,...,t$_n\}$ in Q, find a query decomposition $\gamma(P|BGP, D, T, PTP)$ that satisfies the following conditions:

- The evaluation of $\gamma(P|BGP, D, T, PTP)$ in D is *complete* according to the privacy-aware policies of the federation in T. Suppose D^* represents the maximal subset of D where the privacy policies of each RDF dataset $D_i \in D^*$ allow for projecting and joining the properties from D_i that appear in Q[1]. Then the evaluation of BGP in D^* is equivalent to the evaluation of $\gamma(P|BGP, D, T, PTP)$ and the following expression holds:

$$[[BGP]]_{D*} = [[\gamma(P|BGP, D, T, PTP)]]_D$$

- The cost of executing the query decomposition $\gamma(P|BGP, D, T, PTP)$ is *minimal*. Suppose the execution time of a decomposition P' of BGP in D is represented as $cost(\gamma(P'|BGP, D, T, PTP))$, then

$$\gamma(P|BGP, D, T, PTP) = \underset{\gamma(P'|BGP,D,T,PTP)}{\text{argmin}} cost(\gamma(P'|BGP, D, T, PTP))$$

To solve this problem, we present BOUNCER, a federated query engine able to identify query decompositions for SPARQL queries and query plans that efficiently evaluate SPARQL queries over a federation. Two definitions are presented for a query plan over a decomposition. The next two functions are presented in order to facilitate the understanding of the definition of a query plan.

Definition 24 (The property function prop(*)) *Given a set of triple patterns, TPS, the function $prop(TPS)$ is defined as follows:*

$$prop(TPS) = \{p \mid (s, p, o) \in TPS \ \wedge p \text{ is constant } \}$$

Definition 25 (The variable function var(*)) *Given a privacy-aware decomposition, Φ, the function $var(\Phi)$ is defined inductively as follows:*

1. ***Base case:*** *$\Phi = \{\phi_1\}$, then $var(\Phi) = \{?x \mid (s, p, o) \in SQ_1$, where $\phi_1 = (SQ_1, SD_1, PS_1, PTS_1)$, $?x = s \wedge s$ is a variable $\vee$ $?x = o \wedge o$ is a variable$\}$*

2. ***Inductive case:*** *Let Φ_1 and Φ_2 be disjoint decompositions such that Φ=$\Phi_1 \cup \Phi_2$ then, $var(\Phi) = var(\Phi_1) \cup var(\Phi_2)$.*

Definition 26 (A Valid Plan over a Privacy-Aware Decomposition) *Given a privacy-aware decomposition $\gamma(P|BGP, D, T, PTP)$: $\Phi = \{\phi_1, \ldots, \phi_n\}$, a valid query plan, $\alpha(\Phi)$, is defined inductively as follows:*

1. ***Base Case:*** *If only one decomposition ϕ_1 belongs to Φ, i.e., $\Phi = \{\phi_1\}$, the plan unions all the service graph patterns over the selected RDF sources. Thus, $\alpha(\Phi) = UNION_{d_i \in SD_1}(SERVICE\ d_i\ SQ_1)$ is a valid plan*[2][3]*, where:*
   - *$\phi_1 = (SQ_1, SD_1, PS_1, PTS_1)$ is a valid privacy-aware decomposition;*
   - *All the variables projected in the query have the permission to be projected, i.e., $\forall p_{i1} \in prop(PTS_1)$, $project(Di, pi1, Ci1) \in PS_1$.*

2. ***Inductive Case:*** *Let Φ_1 and Φ_2 be disjoint decompositions such that Φ=$\Phi_1 \cup \Phi_2$. Then, $\alpha(\Phi) = (\alpha(\Phi_1) * \alpha(\Phi_2))$ is a valid plan, where:*

[1] Predicates $project(Di, p_{ij}, C_{ij})$, $join\_fed(Di, p_{ij}, C_{ij})$ and $join\_local(Di, p_{ij}, C_{ij})$ are part of T for all properties in triple patterns in Q that can be answered by Di.

[2] For readability, UNION$_{di \in SD+i}$ represents SPARQL UNION operator

[3] $SERVICE$ corresponds to the SPARQL SERVICE clause

a) $\alpha(\Phi_1)$ *and* $\alpha(\Phi_2)$ *are valid plans.*

b) The join variables appear jointly in the triple patterns of Φ_1 *and* Φ_2*, i.e.,* $joinVars = var(\Phi_1) \cap var(\Phi_2)$.

c) $\mathcal{J}$ *is a set of joint triple patterns involving join variables in BGP:*

- $\mathcal{J} = \{t | variable(t) \subseteq joinVars, (t \in \Phi_{1(SQ)} \lor t \in \Phi_{2(SQ)})\}$
- $\Phi_{1(SQ)} = \{SQ_i | \forall\phi_i \in \Phi_1, \phi_i = (SQ_i, SD_i, PS_i, PTS_i)\}$*, and*
- $\Phi_{2(SQ)} = \{SQ_j | \forall\phi_j \in \Phi_2, \phi_j = (SQ_j, SD_j, PS_j, PTS_j)\}$.

*d) The operator * is a JOIN operator, i.e.,* $\alpha(\Phi)$ *=(*$\alpha(\Phi_1)$ *JOIN* $\alpha(\Phi_2)$*) is a valid plan, iff* $\forall p_{ij} \in prop(\mathcal{J})$, $join\_fed(D_i, p_{ij}, C_{ij}) \in (\Phi_{1(PS)} \cap \Phi_{2(PS)})$, $\Phi_{1(PS)} = \{PS_i | \forall\phi_i \in \Phi_1, \phi_i = (SQ_i, SD_i, PS_i, PTS_i)\}$*, and* $\Phi_{2(PS)} = \{PS_j | \forall\phi_j \in \Phi_2, \phi_j = (SQ_j, SD_j, PS_j, PTS_j)\}$.

*e) The operator * is a DJOIN operator, i.e.,* $\alpha(\Phi)$ *=(*$\alpha(\Phi_1)$ *DJOIN* $\alpha(\Phi_2)$*) is a valid plan iff* $\forall p_{ij} \in prop(\mathcal{J})$, $join\_fed(D_i, p_{ij}, C_{ij}) \in \Phi_{1(PS)}$ *and* $join\_local(D_i, p_{ij}, C_{ij}) \in \Phi_{2(PS)}$[4].

Next, we define the BOUNCER architecture and the main characteristics of the query decomposition and execution tasks implemented by BOUNCER.

7.3 BOUNCER: A Privacy-Aware Query Engine

This section presents the privacy-aware techniques implemented by BOUNCER. They rely on the description of the RDF datasets of a federation in terms of privacy-aware RDF molecule templates (PRDF-MTs) to identify query plans that enforce data access control regulations. More importantly, these techniques are able to generate query execution plans whose operators force the execution of queries at the dataset sites in case data cannot be transferred or accessed.

Definition 27 (Privacy-Aware RDF Molecule Template(PRDF-MT)) *A privacy-aware RDF molecule template (PRDF-MT) is a 5-tuple=<WebI, C, DTP, IntraL, InterL>, where:*

- *WebI – is a Web service API that provides access to an RDF dataset G via SPARQL protocol;*
- *C – is an RDF class such that the triple pattern (?s rdf:type C) is true in G;*
- *DTP – is a set of triples (p, T, op) such that p is a property with domain C and range T, the triple patterns (?s p ?o) and (?o rdf:type T) and (?s rdf:type C) are true in G, and op is an access control operator that is allowed to be performed on property p;*
- *IntraL – is a set of pairs (p,*C_j*) such that p is an object property with domain C and range* C_j*, and the triple patterns (?s p ?o) and (?o rdf:type* C_j*) and (?s rdf:type C) are true in G;*
- *InterL – is a set of triples (p,*C_k*,SW) such that p is an object property with domain C and range* C_k*; SW is a Web service API that provides access to an RDF dataset K, and the triple patterns (?s p ?o) and (?s rdf:type C) are true in G, and the triple pattern (?o rdf:type* C_k*) is true in K.*

7.3.1 Privacy-Aware Source Selection and Decomposition

The BOUNCER privacy-aware source selection and query decomposition is sketched in Algorithm Algorithm 5. Given a *BGP* in a SPARQL query *Q*, BOUNCER first decomposes the query into star-shaped subqueries (SSQs), (Line 2). For instance, our running example query, in Figure7.4a, is decomposed into

[4] DJOIN- is a dependent JOIN [47].

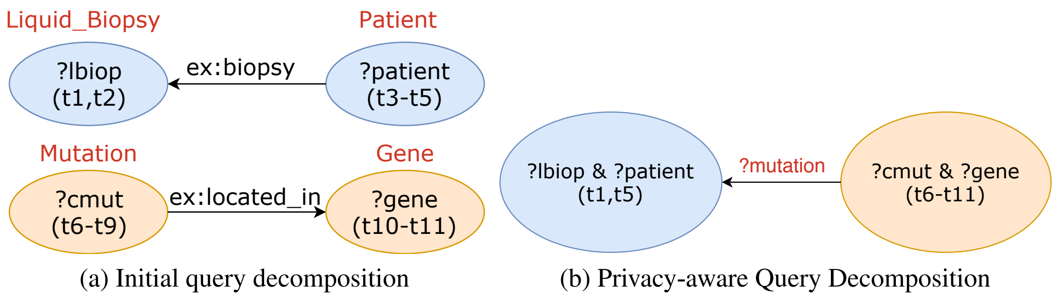


Figure 7.5: **Example of Privacy-Aware Decompositions**. Decompositions for SPARQL query in the motivating example. Nodes represent SSQs and colors indicate datasets where they are executed; edges correspond to join variables. a) Initial query decomposed into four SSQs. b) Decomposition result where the subqueries `?lbiop`-SSQ and `?patient`-SSQ are composed into a single subquery to comply with the privacy policy of data source S1, while `?cmut`-SSQ and `?gene`-SSQ are also composed to push down the join operation to the data source S2.

four SSQs, as shown in Figure7.5, i.e., SSQs around the variables `?lbiop`, `?patient`, `?cmut`, and `?gene`, respectively. The first SSQ (denoted `?lbiop`-SSQ) has two triple patterns, t1-t2, the second SSQ (`?patient`-SSQ) is composed of three triple patterns, t3-t5, the third SSQ (`?cmut`-SSQ) includes four triple patterns, and the fourth SSQ (`?gene`-SSQ) is composed of two triple patterns, t10-t11.

Figure7.5a presents an initial decomposition with the selected PRDF-MTs for each SSQs. The subquery `?patient`-SSQ is joined to the subquery `?lbiop`-SSQ via `ex:biopsy` property. Similarly, `?cmut`-SSQ is joined to `?gene`-SSQ via the `ex:located_in` property. Given the set of properties in each SSQ and the joins between them, BOUNCER finds a matching PRDF-MT for each SSQs (Line 3), i.e., it matches the subqueries `?patient`-SSQ, `?lbiop`-SSQ, `?cmut`-SSQ, and `?gene`-SSQ to the PRDF-MTs `Patient`, `Liquid_Biopsy`, `Mutation`, and `Gene`, respectively. Once the PRDF-MTs are identified for the SSQs, BOUNCER verifies the access control policies associated with them (Line 4). A subquery SSQ associated with an PRDF-MT(s) that grants the `project()` permission to all of its properties is called *Independent SSQ*; otherwise, it is called *Dependent SSQ*. An SSQ in a SPARQL query Q is called *dependent* iff a property of at least one triple pattern in SSQ is associated with the privacy-aware operation `join_local()`. On the other hand, an SSQ is *independent* iff the privacy-aware operation `project()` is true for the properties of the triple patterns in SSQ.

If the value of the controlled property is in the projection list, i.e., if the property of a triple pattern in an SSQ have `join_local()` or `join_fed()` predicate, then the decomposition process exits with empty result (Line 6). Once the SSQs are associated with PRDF-MTs, the next step is to merge the SSQs with the same source and push down the join operation to the data source. To comply with access control policies of a dataset, i.e., when the properties of an SSQ have only the `join_local()` permission, the

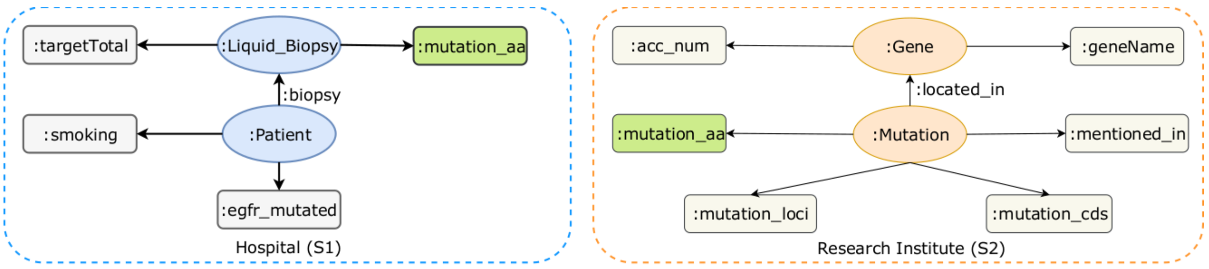


Figure 7.6: **Example of Privacy-aware RDF Molecule Templates (PRDF-MTs)**. Two PRDF-MTs for the SPARQL query in the motivating example. According to the privacy regulations the properties **:name**, **:birthdate**, and **:addresss** are controlled; they do not appear in the PRDF-MTs.

Algorithm 5 Privacy-Aware Query Decomposition: *BG* - Basic Graph Pattern, *Q* - Query, *PRMT* - Access-aware RDF Molecule Templates

```
procedure DECOMPOSE($BGP$, $Q$, $PRMT$)
    $SSQs \leftarrow getSSQs(BGP)$                                ▷ Partition the BGP to SSQs
    $RES \leftarrow selectSource(PRMT, PRMT)$                     ▷ RES=[(SSQ, PRMT, DataSource)]
    $A \leftarrow getAccessPolicies(RES)$; $\Phi \leftarrow [\,]$; $DR \leftarrow \{\,\}$      ▷ access control statements
    for $(SSQ, RMT, p, ds, pred) \in A$ do
        if $p \in Query.PRJ \wedge pred \;!= project(ds, p, RMT.type)$ then return [ ]
        $DR[SSQ][PTS].append(t) \mid t = (s, p, o) \wedge t \in SSQ \mid p \in Query.PRJ$
        $DR[SSQ][SD].append(ds) \wedge DR[SSQ][PS].append(pred)$
    end for
    for $(SSQ_i, SD_i, PS_i, PTS_i) \in DR$ do
        $\phi_i = (SQ_i, SD_i, PS_i, PTS_i) \mid SQ_i \leftarrow SSQ_i$
        if $join\_local() \in PS_i$ then                          ▷ If $SSQ_i$ contains restricted property
            for $(SSQ_j, SD_j, PS_j, PTS_j) \in DR$ do
                if $SD_i \cap SD_j \neq \emptyset$ then
                    $\phi_i.extend(SSQ_j, SD_j, PS_j, PTS_j)$
                    $DR.remove((SSQ_j, SD_j, PS_j, PTS_j)) \wedge done \leftarrow True$
            end for
            if $NOT\ done$ then return [ ]
        end if
        $\Phi.append(\phi_i)$
    end for
    return $\Phi$                                                 ▷ decomposed query
end procedure
```

join operation with this SSQ should be done at the data source. Hence, if two SSQs can be executed at the same source, then BOUNCER decomposes them as a single subquery (SQ) (Line 10-21). This technique may also improve query execution time by performing join operation at the source site. Figure7.5b shows a final decomposition for our running example. `?lbiop`-SSQ and `?patient`-SSQ are merged because they are dependent and the join operation can be executed at the source.

7.3.2 Privacy-Aware Query Planning Technique

Algorithm 6 sketches the BOUNCER privacy-aware query planing technique. Given a privacy-aware decomposition Φ of a query Q, BOUNCER finds a valid plan that respects the privacy-policy of the data sources. For each subquery in ϕ_i a service-graph pattern is created (Line 4 & 6) and the SPARQL UNION operator is used whenever the subquery can be executed over more than one data source. Then, BOUNCER selects another subquery, ϕ_j that is joinable with ϕ_i (Line 5). If ϕ_i is composed of dependent SSQ(s) (resp., independent SSQ(s)) and ϕ_j is composed of an independent SSQ(s) (resp., dependent SSQ(s)), then a dependent join operator (DJOIN) is selected (Line 9-12). If both ϕ_i and ϕ_j are merged of an independent SSQ(s), then any JOIN operator can be chosen (Line 13-14). Finally, otherwise, an empty plan is returned indicating that there is no valid plan for the input query (Line 16).

Algorithm 6 Query Planning over Privacy-Aware Decomposition: Φ - Privacy-Aware query decomposition, Q - SELECT query

```
1: procedure MAKEPLAN($\Phi$, $Q$)
2:     $\alpha \leftarrow []$
3:     for $\phi_i \in \Phi$ do
4:         $\sigma_1 \leftarrow UNION_{d_i \in SD_i \wedge SD_i \in \phi_i}(SERVICE\ d_i\ \ SQ_i)$
5:         for $\phi_j \in \Phi \mid \phi_i \neq \phi_j \wedge var(SQ_i)\ \cap\ var(SQ_j)\ \not\equiv\ \emptyset$ do    ▷ If joinable
6:             $\sigma_2 \leftarrow UNION_{d_j \in SD_j}(SERVICE\ d_j\ \ SQ_j)$
7:             $\mathcal{J} \leftarrow \{\, t \mid vari(t) \subseteq [var(SQ_i) \cap var(SQ_j)] \wedge t \in [SQ_i \cup SQ_j]\}$
8:             $\rho \leftarrow prop(\mathcal{J})$    ▷ Properties of join variables
9:             if $\exists join\_local() \in PS_i \wedge \forall pred_{p \in \rho} \in PS_j \mid pred_{p \in \rho} \Rightarrow join\_fed()$ then
10:                $\alpha.append((\sigma_2\ \ DJOIN\ \ \sigma_1))$; $joined \leftarrow True$    ▷ Dependent JOIN
11:            if $\exists join\_local() \in PS_j \wedge \forall pred_{p \in \rho} \in PS_i \mid pred_{p \in \rho} \Rightarrow join\_fed()$ then
12:                $\alpha.append((\sigma_1\ \ DJOIN\ \ \sigma_2))$; $joined \leftarrow True$    ▷ Dependent JOIN
13:            if $\forall pred_{p \in \rho} \in [PS_i \cup PS_j] \mid pred_{p \in \rho} \Rightarrow join\_fed()$ then
14:                $\alpha.append((\sigma_1\ \ JOIN\ \ \sigma_2))$; $joined \leftarrow True$    ▷ Independent JOIN
15:        end for
16:        if $\exists join\_local() \in PS_i \wedge\ NOT\ \ joined$ then return [ ]    ▷ No valid plan
17:    end for
18:    return $\alpha$
19: end procedure
```

7.3.3 The BOUNCER Architecture

Figure7.7 depicts BOUNCER architecture. Given a SPARQL query, the source selection and query decomposition component solves the problem of identifying a privacy-aware query decomposition; they select PRDF-MTs for subqueries (SSQs) by consulting PRDF-MT metadata store and the access control evaluator component. The source selection and decomposition component is privacy-aware decomposition; it is given to the query planning component for creating a valid plan, i.e., access policies of the selected data sources should be respected. The valid plan is executed in a bushy-tree fashion by the query execution.

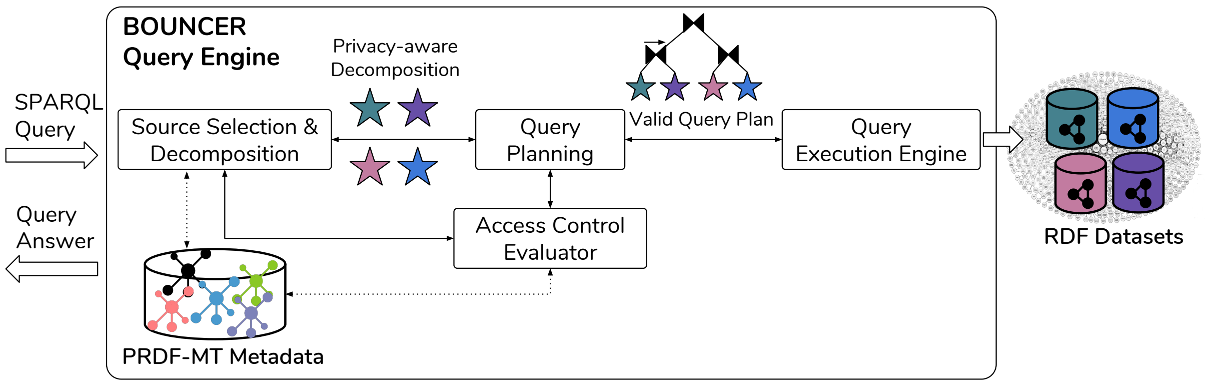


Figure 7.7: **BOUNCER Architecture**. BOUNCER receives a SPARQL query and outputs the results of executing the SPARQL query over a federation of SPARQL endpoints. It relies on PRDF-MT descriptions and privacy-aware policies to select relevant sources, and perform query decomposition and planning. The query engine executes a valid plan against the selected sources.

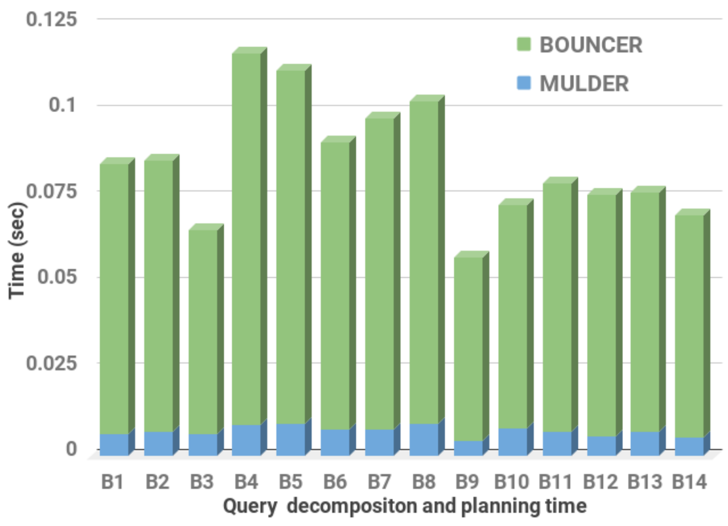


Figure 7.8: **Decomposition and Planning Time.** BOUNCER decomposition and planning are more expensive than baseline (MULDER)

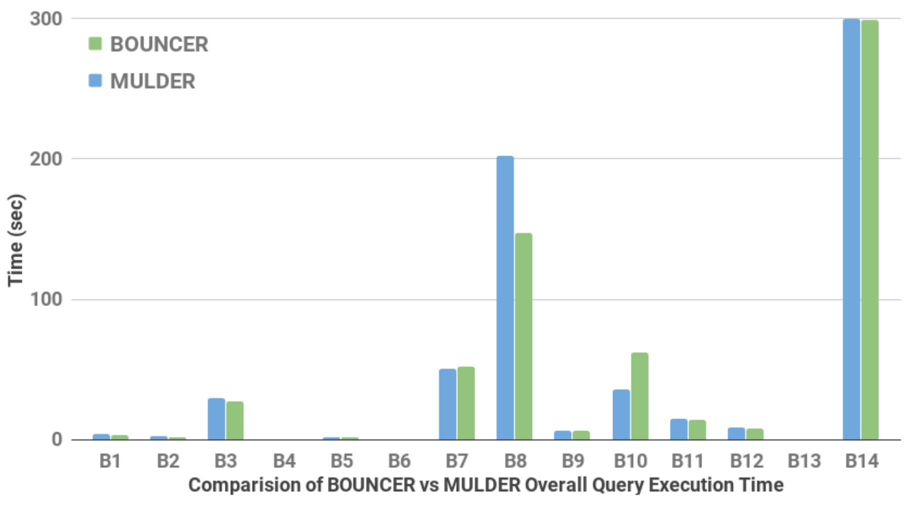


Figure 7.9: **Overall Execution Time**. BOUNCER generates more efficient plans and overall execution time is reduced.

7.4 Empirical Evaluation

We study the efficiency and effectiveness of BOUNCER. First, we assess the impact of access-control policies enforcement and BOUNCER is compared to ANAPSID, FedX, and MULDER. Then, the performance of BOUNCER is evaluated. For our evaluation the following research questions are studied: **Q1)** Does privacy-aware enforcement employed during source selection, query decomposition, and planning impact query execution time? **Q2)** Can privacy-aware policies be used to identify query plans that enhance execution time and answer completeness?
Benchmarks: The Berlin SPARQL Benchmark (*BSBM*) generates a dataset of 200M triples and 14 queries; answer size is limited to 10,000 per query.
Metrics: *i*) `Execution Time`: Elapsed time between the submission of a query to an engine and the delivery of the answers. Timeout is set to 300 seconds. *ii*) `Throughput`: Number of answers produced per second; this is computed as the ratio of the number of answers to execution time in seconds.

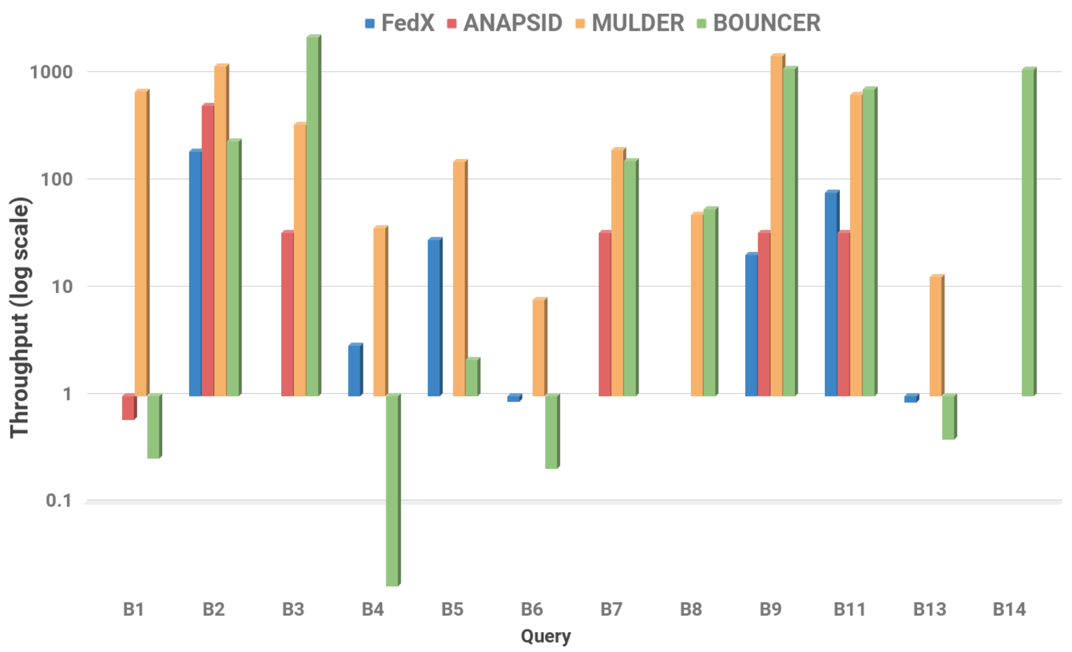


Figure 7.10: **Efficiency of Query Plans**. Existing engines are compared based on throughput. ANAPSID plans are efficient but no valid. FedX and MULDER generate valid plans (by chance) but some are not efficient. BOUNCER generates both valid and efficient plans and overall execution time is reduced.

Implementation: BOUNCER privacy-aware techniques are implemented in Python 3.5 and integrated into the ANAPSID query engine. The BSBM dataset is partitioned into 8 parts (one part per RDF type) and deployed on one machine as SPARQL endpoints using Virtuoso 6.01.3127, where each dataset resides in a dedicated Virtuoso docker container. Experiments are executed on a Dell PowerEdge R805 server, AMD Opteron 2.4GHz CPU, 64 cores, 256GB RAM.

7.4.1 Impact of Access Control Enforcement.

The impact of privacy-aware processing techniques is studied, as well as the overhead on source selection, decomposition, and execution. In this experiment, the privacy-aware theory enables all the operations over the properties of the federation, i.e., all the operations are defined for each property and dataset. MULDER and BOUNCER are compared; Figure7.8 and Figure7.9 reports on decomposition, planning, and execution time per query. Both engines generate the same results and BOUNCER consumes more time in query decomposition and planning. However, the overall execution time is lower in almost all queries. These results suggest that even there is an impact on query processing, BOUNCER is able to exploit privacy-aware polices, and generates query plans that speed up query execution.

7.4.2 Impact of Privacy-Aware Query Plans.

The privacy-aware query plans produced by BOUNCER are compared to the ones generated by state-of-the-art query engines. In this experiment, the privacy-aware theory enables local joins for `Person`, `Producer`, `Product`, and `ProductFeature`, and projections of the properties of `Offer`, `Review`, `ProductType`, and `Vendor`. Figure7.10 reports on the throughput of each query engine. As observed, the query engines produced different query plans which allow for high performance. However, many of these plans are not valid, i.e., they do not respect the privacy-aware policies in the theory. For instance, ANAPSID produces bushy tree plans around `gjoins`; albeit efficient, these plans violate the privacy policies. FedX and MULDER are able to generate some valid plans–*by chance*– but fail in producing efficient executions. On the contrary, BOUNCER generates valid plans that in many cases increase the performance of the query engine. Results observed in two experiments suggest that efficient query plans can be identified by exploiting the privacy policies; thus, **Q1** and **Q2** can be positively answered.

7.5 Summary

We presented BOUNCER, a privacy-aware federated query engine for SPARQL endpoints. BOUNCER relies on privacy-aware RDF Molecule Templates (RDF-MTs) for source description and guiding query decomposition and plan generation. Privacy-aware RDF-MTs are able to encode different privacy-aware operations that are allowed to be performed at different level of processing. The proposed privacy-aware query decomposition and planning algorithms are able to generate a valid plan according to the allowed operations that can be performed either at the local sources, federation engine, or projected to users. Efficiency of BOUNCER was empirically evaluated and results suggest that it is able to reduce query execution time and increase answer completeness by producing query plans that comply with the privacy policies of the data sources.

CHAPTER 8

Interest-based Update Propagation

In recent years, there has been an increasing number of structured data published on the Web as a Linked Open Data (LOD). As of March 2019, the LOD cloud[1] comprises of **1,239** datasets with **16,147** links between them. Many of these datasets, such as DBpedia and Wikidata, are voluminous and process large amount of requests from diverse applications. Providing services on top of these datasets is becoming a challenge due to the lack of service levels regarding the availability of datasets [104] and restrictions imposed by the publisher on the type of query forms and number of results[2]. Many data products and services rely on full or partial local LOD replications to ensure faster querying and processing. While such replicas enhance the flexibility of information sharing and integration infrastructures, they also introduce data duplication with all the associated undesirable consequences. Given the evolving nature of the original and authoritative datasets, to ensure consistent and up-to-date replicas, frequent replacements are required at a great cost. In this chapter, we tackle one of the challenges maintaining the freshness of data sources in a Semantic Data Lake by propagating updates from original data provider in a selective and efficient way. The content of this chapter is based on the publications [105–107]. The result of this chapter provides answer to the following research question:

RQ4: How can we define update interests and propagate interesting updates for manage (co)evolution of data sources?

To answer this research question, we study the techniques for propagating updates from original to target data source and the synchronization problems that may be generated if target data source are also allowed to evolve over time. The target data sources might be a replica of the whole dataset or only a slice of the dataset that the application is interested in. We introduce an approach for *interest-based update propagation*, which propagates only interesting parts of updates from the original to the target data source and vice-versa. Effectively, this enables remote applications to 'subscribe' to relevant data sources and consistently reflect the necessary changes locally without the need to frequently replace the entire dataset (or a relevant subset) as well as they can also offer their updates propagated back to the original data source. In this chapter, we use the terms *data source* and *dataset* interchangeable. Our approach is based on a formal definition for graph-pattern-based interest expressions that is used to filter interesting parts of updates from the source. We define a technique to describe an interest for update propagation based on basic graph pattern (BGP) expressions of the SPARQL query. Such interest expressions extend the source description model, RDF Molecule Template (Chapter 4).

[1] https://lod-cloud.net/
[2] https://lists.w3.org/Archives/Public/public-lod/2011Aug/0028.html

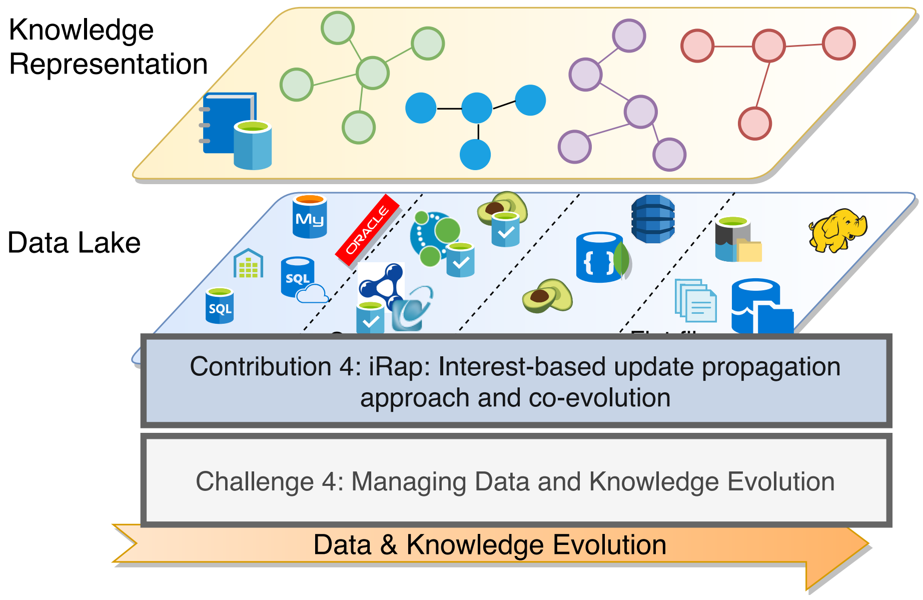

Knowledge Representation
Data Lake
Contribution 4: iRap: Interest-based update propagation approach and co-evolution
Challenge 4: Managing Data and Knowledge Evolution
Data & Knowledge Evolution

Figure 8.1: **Challenges and Contributions**.This chapter focuses on the problem of privacy and access control federated query processing.

We propose a co-evolution approach comprises of the following components: (a) an RDF data synchronization component, and (b) a component for conflict identification and resolution. We implement the approach in the **iRap** framework and perform a comprehensive evaluation based on DBpedia Live updates, to confirm the validity and value of our approach. Our approach relies on the *assumption* that either the data source provider provides a tool to compute a changeset at real-time or third party tools can be used for this purpose. Another *assumption* is that *slices* of the RDF data from the source dataset are replicated, i.e., *target datasets*, where a slice[3] corresponds to an RDF subgraph of the source RDF graph [59]. Our evaluation shows, that the data required to be transferred and handled by applications can be reduced by several orders of magnitude thus substantially lowering the re-usage barrier for Linked Data. In addition, experimental results suggest that our synchronization, and conflict identification and resolution techniques positively affect the quality of the data in both the source and target datasets. Figure 8.1 shows the challenge tackled in this chapter and the contribution of in this chapter.

The contribution of this chapter can be summarized as follows:

- A thorough formalization of interest-based update propagation approach.
- An RDF update propagation framework, **iRap**.
- A conflict detection and resolution strategy
- An empirical evaluation based on DBpedia Live updates to confirm the validity and value of our approach. The results show that the data required to be transferred and handled by applications can be reduced by several orders of magnitude thus substantially lowering the re-usage barrier for Linked Data.

The article is structured as follows: First, in Section 8.1 we motivate the problem of update propagation and co-evolution. Section 8.3 extensively describes the formalization for our framework. Section 8.5

[3] An RDF slice is also known as a fragment in the approaches proposed by Ibañez et al. [108], Montoya et al. [60], and Verborgh et al. [104].

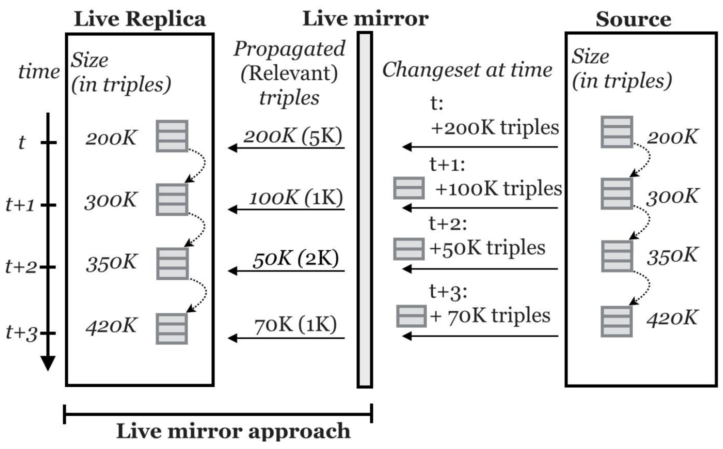


Figure 8.2: Live mirror based changeset propagation approach

and Section 8.6 discusses the implementation and evaluation of the iRap framework in detail. Finally, Section 8.7 present closing remarks of this chapter.

8.1 Motivation

Replication of Linked Data datasets enhances flexibility of information sharing and integration infrastructures. Since hosting a replica of large datasets is costly, organizations might want to host only a relevant subset of the data, for example, using approaches such as *RDFSlice* [109]. However, due to the evolving nature of these datasets in terms of content and ontology, maintaining a consistent and up-to-date replica of the relevant data is a major challenge. Resources in a dataset might be added, updated, or removed. Applications consuming these datasets should be capable of dealing with such updates to keep their local copies consistent.

Let us assume a mobile application which requires information of restaurants (i.e., name, rating, chef, description, and depiction) nearby users' location. DBpedia dataset can be used to find such information. A mirror (replica) of the dataset need to be created as the application requires a high query performance without restrictions imposed by DBpedia public endpoint [4]. The replicated dataset might undergo changes as the mobile application supports features such as allowing users to give reviews about their experience in those restaurants. At the same time, DBpedia dataset also evolves by adding new restaurants information or updating the existing ones. As a result, the target dataset might be out of date and need to be synchronized with DBpedia. Typically, dataset mirror applications propagate updates published by the source dataset to a target dataset. For instance, the *DBpedia Live mirror tool*[5] propagates all changes to a target dataset, so that at any point of time the target dataset contains the same triples as the DBpedia Live dataset. However, for example, an application interested in only restaurants uses only $1,950$ out of $6.6M$ instances of the English DBpedia 2016 dataset[6]. As a result, the target dataset will grow as the source dataset, even though, the fragment of data the application is interested in have not changed as much as other fragments in the data source.

Figure 8.2 shows the propagation of unfiltered data from a source to a target, referred to as *Live Replica*). This approach propagates all the updates irrespective of the relevance or usefulness of the data for the target application. As can be seen, using the *Live Replica* initially it contains $200K$ triples, same

[4] http://dbpedia.org/sparql
[5] https://github.com/dbpedia/dbpedia-live-mirror
[6] https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10

as the source dataset, even though the target application is interested only $5K$ triples. Additionally, the *Live Replica* grows by $220K$ more triples while the interesting updates only grows by $4K$ triples. In addition, if the target dataset is allowed to evolve, synchronization of updates between the source and target might become even more challenging, as different inconsistency (data conflicts) might occur.

In this chapter, we present an approach for interest-based update propagation, which is based on the specification of data interests by a target application. Based on such interest expressions all updates are evaluated and only those changes satisfying target applications' interest are shipped to the target dataset. An interest-based update propagation could significantly reduce the amount of data to be shipped and managed at the application side and thus lower the barrier for the deployment of Linked Data applications. In addition, we tackle the problem of co-evolution when the target dataset is allowed to evolve and device an approach for conflict resolution during synchronization.

8.2 Problem Statement and Proposed Solution

In this section, we define the co-evolution problem when a slice or replica of original source dataset is created and only updates for a selected parts of a slice or replica is required to co-evolve with the original data sources. We present our proposed solution to solve the co-evolution problem.

8.2.1 Problem Statement

The core of the co-evolution concept relies on the mutual propagation of changes between the source and target datasets in order to keep the datasets *in sync*. Thus, from time to time, the source and target datasets have to exchange the changesets and then update the local repositories. Issues are about how changes shWe propose a two-fold co-evolution approach, comprised of the following components: *i*) an RDF data synchronization component, and *ii*) a component for conflict identification and resolution. ould be propagated and in case of *inconsistencies* or *data conflicts*, how these conflicts should be resolved. Thus, our main research problem is to develop a co-evolution process able to exploit the properties of RDF data and solve conflicts generated by the propagation of changes among source datasets and replicas.

Given source dataset, S, target dataset, T, and (interest) query expression, I, the problem of co-evolution of target dataset T with respect to source dataset, S, under query expression, I, is defined as follows. For all BGPs in I, find a synchronization strategy that satisfies the following requirements:

- **Initial Inclusion:** At the initial time, t_0, the target dataset, T, is a subset of the source dataset, S, under restriction query, I, $T_{t_0} \subseteq_I S_{t_0}$, i.e., $[[I]]_{T_{t_0}} = [[I]]_{S_{t_0}}$;
- **Inclusion after Synchronization:** At any time, t_i, after initial synchronization, the target dataset should be a subset of the source dataset, $T_{t_i} \subseteq_I S_{t_i}$, i.e., $[[I]]_{T_{t_i}} = [[I]]_{S_{t_i}}$.

8.2.2 Proposed Solution

To solve the co-evolution problem, we propose an interest-based update propagation, that propagates only selected set of triple from source to target dataset or vice-versa. We propose a two-fold co-evolution approach, comprised of the following components: *i*) an RDF data synchronization component, and *ii*) a component for conflict identification and resolution. We propose different synchronization strategies based on the type of requirements by the target application. To resolve any conflicts that might occur during synchronization, we propose different conflict identification and resolution policies and functions. We implement our proposed solution in iRap update propagation framework and integrated the conflict identification and resolution policies.

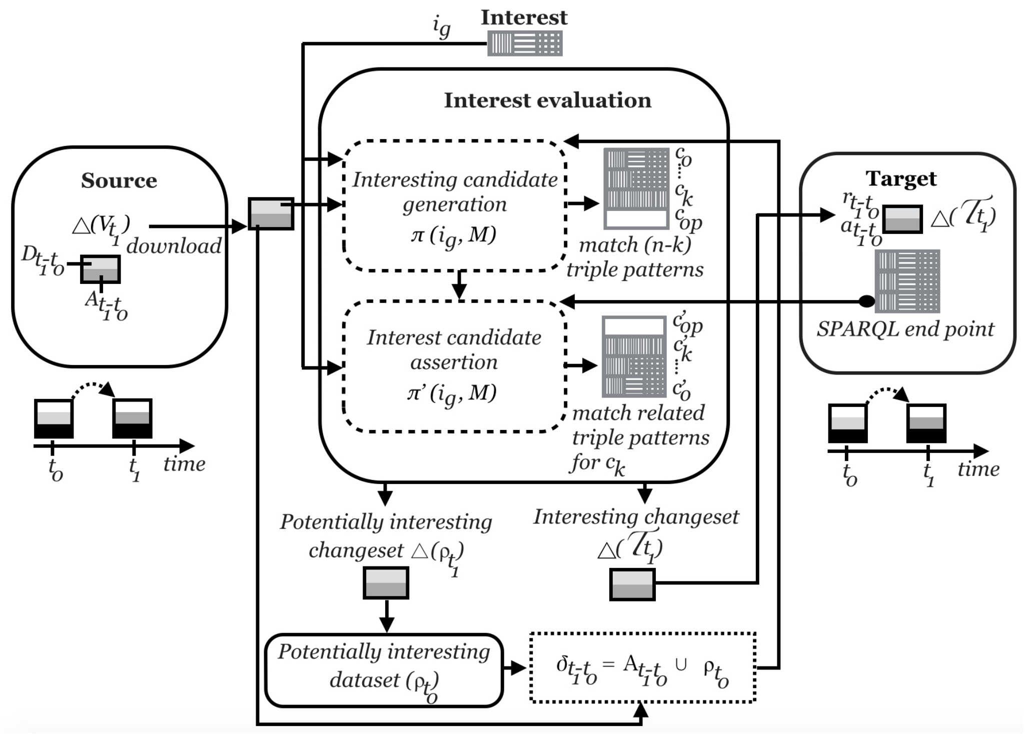

Interest
i_g
Interest evaluation
Source
$\triangle(V_{t_1})$
download
$D_{t_1-t_0}$
$A_{t_1-t_0}$
Interesting candidate generation
$\pi\,(i_g, M)$
c_0
c_k
c_{op}
match (n-k) triple patterns
Interest candidate assertion
$\pi'\,(i_g, M)$
c'_{op}
c'_k
c'_0
match related triple patterns for c_k
Target
$r_{t_1-t_0}$
$a_{t_1-t_0}$
$\triangle(\mathcal{I}_{t_1})$
SPARQL end point
t_0
t_1
time
Potentially interesting changeset $\triangle(\rho_{t_1})$
Interesting changeset $\triangle(\mathcal{I}_{t_1})$
Potentially interesting dataset (ρ_{t_0})
$\delta_{t_1-t_0} = A_{t_1-t_0} \cup \rho_{t_0}$

Figure 8.3: Formalization overview of the interest-based update propagation.

8.3 Formalization of Interest-based Update Propagation

In this section, we present the formal description of our interest-based update propagation approach. Figure 8.3 illustrates the overall interest-based Update Propagation approach; summarizing the concepts defined through the formalization. Interest evaluation takes place over the input set of deleted $(D_{t_1-t_0})$ and added $(At_1 - t_0)$ triples from the source dataset (V_{t_1}) in between time interval (t_0, t_1). Since updates do not only contain interesting and uninteresting parts, but also triples which can become potentially interesting along with subsequent updates. We have to compute and store these sets of potentially interesting triples and take them in subsequent update assessments into account.

For our formalization we will use the notations **U**, **B**, **L** and *Var* for the disjoint sets of all IRIs, blank nodes, literals (typed and untyped) and variables respectively. An *RDF graph* V is a finite set of RDF triples, i.e, $V \subset (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$. In what follows, we use the terms RDF graph, *RDF dataset*, and *dataset* interchangeably. An *evolving dataset*, V^g, is a dataset identified using the persistent IRI, g, whose content changes over time. V_t^g denotes a specific revision of V^g at a particular time t. For simplicity, we will just refer to V_t instead of V_t^g.

Definition 28 (Non-disjoint BGP) *A non-disjoint BGP is a BGP that represents a connected graph.*

An optional graph pattern (OGP) is syntactically specified with the OPTIONAL keyword applied to a graph pattern. A set of triple patterns in a BGP must match for there to be a solution whereas triple patterns in OGP may extend the solution but their non-binding nature means that they cannot reject it [110].

Definition 29 (Partial Matches) *Partial matches are a set of triples that does not fully match the BGP but matches at least one triple pattern in BGP or OGP of a query.*

```
dbr:Marcel   dbp:goals   1 .
dbr:Marcel   dbo:team    dbr:FNFT .
dbr:Tim_Berners-Lee foaf:name
           "Tim Berners-Lee" .
dbr:Cristiano_Ronaldo dbo:goals 96.
```

Listing 8.1: File 000001.removed.nt

```
dbr:Cristiano_Ronaldo  dbo:goals 216 .
dbr:Barack_Obama     foaf:name
                  "Barack Obama" .
dbr:Barack_Obama foaf:homepage
     "http://www.barackobama.com/" .
dbr:Rio_Ferdinand  a     foaf:Person .
dbr:Rio_Ferdinand  a     dbo:Athlete .
dbr:Rio_Ferdinand  dbp:goals 2 .
dbr:Arvid_Smit     a     dbo:Athlete .
```

Listing 8.2: File 000001.added.nt

Triples added to, and removed from, an evolving dataset within a time-frame are called *changeset* for a dataset within that time-frame.

Definition 30 (Changeset) *Let V_{t_1} be an evolving dataset at time t_1. A changeset $\Delta(V_{t_1-t_0})$, between V_{t_0} and V_{t_1}, where $t_0 < t_1$, is defined as:*

$$\Delta(V_{t_1-t_0}) = \langle D_{t_1-t_0}, A_{t_1-t_0} \rangle$$

where:

- *$D_{t_1-t_0}$ is a set of removed triples from V_{t_0} between time-points t_0 and t_1, and*
- *$A_{t_1-t_0}$ is a set of added triples to V_{t_0} between time-points t_0 and t_1.*

Changesets can be computed using the difference between two versions of the RDF dataset. The result of this computation gives the removed triples, $D_{t_1-t_0} = V_0 \backslash V_1$, and added triples, $A_{t_1-t_0} = V_1 \backslash V_0$, between given dataset revisions V_{t_0} and V_{t_1}. Datasets can be accompanied with a tool that publishes changesets at real-time, so that users can download these changesets and synchronize their local replicas. For instance, DBpedia publishes updates in a public changesets folder [7].

Example 6 *Let us assume two files (Listing 8.1 and Listing 8.2) are being published by the DBpedia Live extractor for the changes made on Feb 06, 2015 between 05:00 PM (t_0) and 05:02 PM (t_1).*

A changeset $\Delta(V_{t_1-t_0})$ for the DBpedia Live dataset between t_0 and t_1, contains $D_{05:02-05:00}$ = 000001.removed.nt and $A_{05:02-05:00}$ = 000001.added.nt.
That is, $\Delta(V_{05:02}) = \langle 000001.removed.nt,\ 000001.added.nt \rangle$.

Definition 31 (Changeset Propagation) *A changeset propagation is a function υ that transforms a given dataset V_{t_0} to a new dataset V_{t_1} by applying a changeset, $\Delta(V_{t_1-t_0})$. That is: $\upsilon(V_{t_0}, \Delta(V_{t_1-t_0})) = (V_{t_0} \backslash D_{t_1-t_0}) \cup A_{t_1-t_0} = V_{t_1}$*

The changeset propagation function υ, for example, deletes the triples in 000001.removed.nt from the target dataset and then inserts all triples from 000001.added.nt. This order of operation (deleted first) ensures that inserted triples are not removed again immediately. If an organization maintaining a replica wants to host only a subset of the original dataset, it needs to obtain only relevant updates for this subset. For that purpose, we specify *interests* to subscribe to 'interesting' changes only. During interest

[7] http://live.dbpedia.org/changesets/

registration, an organization provides information about the source dataset to synchronize with, a target dataset endpoint that supports SPARQL Update to propagate interesting changes, and an interest query to select relevant parts of a changeset.

Definition 32 (Interest Expression) *An interest expression over an evolving dataset, V_t, is defined as: $i_g = \langle \tau, b, op \rangle$ where g is an IRI identifying an evolving RDF dataset V_t, τ is an IRI identifying the target dataset endpoint, b is a non-disjoint BGP, and op is an optional graph pattern (OGP) connected to b.*

Example 7 *An interest expression for a list of an athlete with information about goals scored, and optionally their homepage, is expressed as follows:*

- *g = http://live.dbpedia.org/changesets*
- *τ = http://localhost:3030/target/sparql*
- *b =* `{ ?a a dbo:Athlete . ?a dbp:goals ?goals . }`
- *op =* `{ ?a foaf:homepage ?page . }`

The equivalent interest expression SPARQL query will be:

```
CONSTRUCT
WHERE {
        ?a   a   dbo:Athlete .
        ?a   dbp:goals   ?goals .
        OPTIONAL { ?a   foaf:homepage   ?page . }
      }
```

Evolving data sources are composed of RDF molecules that can change over time. Interest expression can be embedded in descriptions of evolving data sources, referred as Evolving RDF Molecule Templates defined as follows:

Definition 33 (Evolving RDF Molecule Template (ERDF-MT)) *An Evolving RDF Molecule Template (ERDF-MT) is a 5-tuple=<DS,C,DTP,IntraL,InterL>, where:*

- *DS – is a set of tuples (WebI, M, i_G) such that WebI is a URL that provide access to the dataset G, M is a set of mappings from source schema to molecule predicates, DTP, and i_G is an interest expression $i_G = \langle \tau, b, op \rangle$ in which the BGP, b, is grounded by the RDF Class and predicates of this RDF-MT, i.e., C and DTP, respectively;*
- *C – is an RDF class such that the triple pattern (?s T_p C) is true in G, where T_p is a typing predicate such as $rdf : type$, or $wdt : P31$;*
- *DTP – is a set of pairs (p, T) such that p is a property with domain C and range T, and the triple patterns (?s p ?o), and (?s T_p C) are true in G;*
- *IntraL – is a set of pairs (p,C_j) such that p is an object property with domain C and range C_j, and the triple patterns (?s p ?o) and (?o T_p C_j) and (?s T_p C) are true in G;*
- *InterL – is a set of triples (p,C_k,SW) such that p is an object property with domain C and range C_k; SW is a URL that provides access to an dataset K, the triple patterns (?s p ?o) and (?s T_p C) are true in G, and the triple pattern (?o T_p C_k) is true in K.*

In order to initialize a local data store, i.e., the target dataset, SPARQL CONSTRUCT queries can be used by employing the interest expression's BGPs to extract and load a subset of the source dataset. Then interest expressions are registered with our iRap framework to retrieve interesting updates from the source dataset. iRap evaluates interest expressions over changesets being published along with the source dataset. Without a restriction of generality, we assume interest expressions here to be static for the lifetime of a target dataset, since an evolution of interest expressions can be simulated by removal and addition. The result of executing an interest evaluation for an interest expression against a changeset are three sets or triples: *1. interesting, 2. potentially interesting, and 3. uninteresting* triples.

Definition 34 (Interesting Triples) *Interesting triples are all triples comprised in full matches of the BGP and possibly OGP of an interest expression, i_g, against the sets of added or deleted triples of a changeset. Interesting triples originating from the first element (i.e., removed triples ($D_{t_1-t_0}$)) of a changeset, $\Delta(V_{t_1-t_0})$, are called interesting-removed triples. Interesting triples originating from the second element (i.e., added triples ($A_{t_1-t_0}$)) of a changeset, $\Delta(V_{t_1-t_0})$, are called interesting-added triples.*

In addition to parts of an changeset for which the 'interestingness' can be immediately decided, there might also be parts, which are *potentially interesting* since, i) the missing parts to render them as interesting are already contained in the target knowledge base or ii) they will be propagated in subsequent updates.

Definition 35 (Potentially Interesting Triples) *Potentially interesting triples are triples comprised in partial matches of the BGP or in OGP of interest expression, i_g:*

- *Potentially interesting triples originating from the first element (i.e., removed triples ($D_{t_1-t_0}$)) of a changeset $\Delta(V_{t_1-t_0})$, are called potentially interesting-removed triples.*
- *Potentially interesting triples originating from the second element (i.e., added triples ($A_{t_1-t_0}$)) of a changeset, $\Delta(V_{t_1-t_0})$, are called potentially interesting-added triples.*

Potentially interesting triples can become interesting if triples missing in the changeset, but required for a full BGP match, are found in the target dataset or in subsequent changesets. Finally, there are triples in the changeset that are neither interesting nor potentially interesting.

Definition 36 (Uninteresting Triples) *Uninteresting triples are triples that do not match any triple pattern in a BGP or OGP of any interest expression, i_g, against the sets of added or deleted triples of a changeset.*

Uninteresting triples are not interesting at the moment and can never become interesting with subsequent changesets. iRap uses an interest query to select candidate triples from a changeset and to assert from a target dataset. These candidates are retrieved in decreasing order of number of matching BGP triple patterns of interest expressions and triples that match any part of optional graph patterns.

Definition 37 (Interest Candidate Generation) *An interest candidate generation is the extraction of matching triples from a changeset for a non-disjoint combination of triple patterns in BGP of an interest expression, i_g. The result of this extraction is an $(n+1)$-tuple with decreasing order of matching:*

$$\pi(i_g, M) = \langle c_0, c_1, ..., c_{n-1}, c_{op} \rangle$$

where:

- *M is a set of removed (respectively added) triples in a changeset,*
- *n is the number of triple patterns in the BGP of interest expression, i_g,*
- *c_k is a set of candidate triples in M that match $n-k$ $(0 \leqslant k < n)$ triple patterns of the BGP of the interest expression, i_g, and*
- *c_{op} is a set of candidate triples in M that match at least one triple pattern in the OGP of interest expression, i_g, but none of the triple patterns in the BGP.*

Example 8 *An interest candidate generation for the interest expression i_g from Example 7 over the changeset from Example 6 gives the following result:*

1. $\pi(i_g, D_{05:02-05:00}) = \langle c_0, c_1, c_{op} \rangle$ *where:*
   $c_0 = \varnothing$
   $c_1 =$ `dbr:Marcel dbp:goals 1.  dbr:Cristiano_Ronaldo dbo:goals 96.`
   $c_{op} = \varnothing$
2. $\pi(i_g, A_{05:02-05:00}) = \langle c_0, c_1, c_{op} \rangle$ *where:*
   $c_0 =$ `dbr:Rio_Ferdinand a dbo:Athlete .  dbr:Rio_Ferdinand dbp:goals 10.`
   $c_1 =$ `dbr:Cristiano_Ronaldo dbp:goals 216 .`
   `dbr:Arvid_Smit a dbo:Athlete.`
   $c_{op} =$ `dbr:Barack_Obama foaf:homepage "http://www.barackobama.com".`

Now an interest candidate assertion verifies candidate triples with respect to all triple patterns in the BGP of an interest expression.

Definition 38 (Interest Candidate Assertion) *The candidate assertion function extracts missing triples for the candidate, c_i of $\pi(i_g, M)$ of an interest expression i_g from the target dataset, τ_{t_0}:*

$$\pi'(i_g, M) = \langle c'_{op}, c'_{n-1}, ..., c'_1, c'_0 \rangle$$

where:

- *M is a set of removed (respectively added) triples in a changeset,*
- *n is the number of triple patterns in the BGP of interest expression, i_g,*
- *c'_{op} is a set of triples from target dataset, τ, that matches the missing optional graph patterns for candidate c_0, of $\pi(i_g, M)$,*
- *c'_k is a set of triples from target dataset, τ, that matches the missing triple patterns for candidate c_{n-k}, where $0 < k < n$, of $\pi(i_g, M)$, and*
- *c'_0 is a set of triples from target dataset, τ, that matches all triple patterns in BGP of interest expression for candidate c_{op}, of $\pi(i_g, M)$.*

Example 9 *Let the target dataset, τ_{t_0}, at time t_0 contains the following triples:*

```
#Target dataset at time t0 = 05:00 PM Feb 06, 2015
dbr:Marcel               a           dbo:Athlete .
dbr:Marcel               dbp:goals   1 .
dbr:Cristiano_Ronaldo    a        dbo:Athlete .
dbr:Cristiano_Ronaldo    dbo:goals   96 .
dbr:Cristiano_Ronaldo   foaf:homepage   "http://cristianoronaldo.com" .
```

An interest candidate assertion for interest candidates generated in Example 8 yields the following result:

1. $\pi'(i_g, D_{05:02-05:00}) = \langle c'_{op}, c'_1, c'_0 \rangle$ *where:*
   $c'_{op} = \varnothing$
   $c'_1 =$ `dbr:Marcel a dbo:Athlete .`
   `dbr:Cristiano_Ronaldo a dbo:Athlete .`
   `dbr:Cristiano_Ronaldo foaf:homepage`
   `"http://cristianoronaldo.com" .`

$\boldsymbol{c'_0} = \emptyset$

2. $\pi'(i_g, A_{05:02-05:00}) = \langle c'_{op}, c'_1, c'_0 \rangle$ *where:*

$\boldsymbol{c'_{op}} = \emptyset$

$\boldsymbol{c'_1}$ = `dbr:Cristiano_Ronaldo a dbo:Athlete .`
`dbr:Cristiano_Ronaldo foaf:homepage`
`"http://cristianoronaldo.com" .`

$\boldsymbol{c'_0} = \emptyset$

The interest evaluation over a changeset $\Delta(V_{t_1-t_0})$ is performed in two steps. First, interest expressions are evaluated against removed triples of a changeset as $d(i_g, D_{t_1-t_0})$, see Definition 39. Second, interest expressions are evaluated against added triples of a changeset as $\alpha(i_g, A_{t_1-t_0})$, see Definition 40. During interest evaluation, added triples are combined with potentially interesting triples from previous changesets (i.e., $I_{t_1-t_0} = A_{t_1-t_0} \cup \rho_{t_0}$) to check their potential promotion to interesting triples.

Definition 39 (Interest Evaluation over Deleted Triples) *Interest evaluation over deleted triples is a function, $d(i_g, D_{t_1-t_0})$, that returns a 3-element tuple*[8]*:*

$$d(i_g, D_{t_1-t_0}) = \pi(i_g, D_{t_1-t_0}) \ \cup^* \ \pi'(i_g, D_{t_1-t_0}) = \langle r_{t_1-t_0}, r_{i(t_1-t_0)}, r'_{t_1-t_0} \rangle$$

where:

- $\pi(i_g, D_{t_1-t_0})$ *is an interest candidate generation against deleted triples,*
- $\pi'(i_g, D_{t_1-t_0})$ *is an interest candidate assertion against deleted triples,*
- $r_{t_1-t_0} = \{c_0 \cup c_k \cup c_{op} | \ c_0, c_k, c_{op} \in \pi(i_g, D_{t_1-t_0})$ *and* $\exists c'_{n-k}, c'_0 \in \pi'(i_g, D_{t_1-t_0})\}$ *is the set of interesting removed triples, i.e., no longer interesting,*
- $r_{i(t_1-t_0)} = \{c_k \cup c_{op} | c_k, c_{op} \in \pi(i_g, D_{t_1-t_0})$ *and* $\nexists c'_{n-k}, c'_0 \in \pi'(i_g, D_{t_1-t_0})\}$ *is the set of potentially interesting removed triples (existing only in removed triples of a changeset) and*
- $r'_{t_1-t_0} = \{c'_0 \cup c'_k \cup c'_{op} | c'_0, c'_k, c'_{op} \in \pi'(i_g, D_{t_1-t_0})$ *and* $\exists c_{op}, c_{n-k}, c_0 \in \pi(i_g, D_{t_1-t_0})\}$ *is the set of triples that become potentially interesting after removing* $r_{t_1-t_0}$.

Example 10 *An interest evaluation over deleted triples in our running example (using the results of Example 8 and Example 9, respectively) is as follows:*

$$\begin{aligned} d(i_g, D_{05:02-05:00}) &= \pi(i_g, D_{05:02-05:00}) \ \cup^* \ \pi'(i_g, D_{05:02-05:00}) \\ &= \langle r_{05:02-05:00}, r_{i(05:02-05:00)}, r'_{05:02-05:00} \rangle \end{aligned}$$

1. $r_{05:02-05:00} = c_1$ *(in Example 8)*

```
dbr:Marcel               dbp:goals    1 .
dbr:Cristiano_Ronaldo    dbo:goals    96 .
```

2. $r_{i(05:02-05:00)} = \emptyset$ *(Since all the potentially interesting removed triples of c_1 in Example 8 becomes interesting and no other triples in c_op)*
3. $r'_{05:02-05:00} = c'_1$

[8] Note: $\cup^*$ indicates that after the component-wise union of the two sets the results are combined to three categories of the resulting 3-tuple, namely, (i) elements from left that have matching right elements, (ii) elements from left that do not have matching right elements, and (iii) element from right that have a match left.

```
dbr:Marcel               a              dbo:Athlete .
dbr:Cristiano_Ronaldo    a              dbo:Athlete .
dbr:Cristiano_Ronaldo    foaf:homepage  "http://cristianoronaldo.com" .
```

Definition 40 (Interest Evaluation over Added Triples) *Interest evaluation over added triples is a function, $\alpha(i_g, A_{t_1-t_0})$, that returns 3 element tuple as:*

$$\alpha(i_g, A_{t_1-t_0}) = \pi(i_g, I_{t_1-t_0}) \ \cup^* \ \pi'(i_g, I_{t_1-t_0}) = \left\langle a_{t_1-t_0}, a_{i(t_1-t_0)}, a'_{t_1-t_0} \right\rangle$$

where:

- $I_{t_1-t_0} = A_{t_1-t_0} \cup \rho_{t_0}$ *is a set of added triples and potentially interesting triples dataset,*
- $\pi(i_g, I_{t_1-t_0})$ *is an interest candidate generation over* $I_{t_1-t_0}$*,*
- $\pi'(i_g, I_{t_1-t_0})$ *is an interest candidate assertion over* $I_{t_1-t_0}$*,*
- $a_{t_1-t_0} = \{c_0 \cup c_k \cup c_{op} | \ c_0, c_k, c_{op} \in \pi(i_g, I_{t_1-t_0})$ *and* $\exists c'_{n-k}, c'_0 \in \pi'(i_g, I_{t_1-t_0})\}$ *is the set of interesting added triples,*
- $a_{i(t_1-t_0)} = \{c_k \cup c_{op} | c_k, c_{op} \in \pi(i_g, I_{t_1-t_0})$ *and* $\nexists c'_{n-k}, c'_0 \in \pi'(i_g, I_{t_1-t_0})\}$ *is the set of potentially interesting added triples that do not have related triples in target dataset, and*
- $a'_{t_1-t_0} = \{c'_0 \cup c'_k \cup c'_{op} | c'_0, c'_k, c'_{op} \in \pi'(i_g, I_{t_1-t_0})$ *and* $\exists c_{op}, c_{n-k}, c_0 \in \pi(i_g, I_{t_1-t_0})$ *respectively} is the set of triples from target dataset that are related to* $a_{i(t_1-t_0)}$.

Example 11 *An interest evaluation over added triples in our running example (using the results of Example 8 and Example 9, respectively) is as follows:*

$$\begin{aligned} \alpha(i_g, A_{05:02-05:00}) &= \pi(i_g, I_{05:02-05:00}) \ \cup^* \ \pi'(i_g, I_{05:02-05:00}) \\ &= \left\langle a_{05:02-05:00}, a_{i(05:02-05:00)}, a'_{05:02-05:00} \right\rangle \end{aligned}$$

1. $a_{05:02-05:00} = c_1 \cup c'_1 \cup c_0$

```
dbr:Cristiano_Ronaldo    dbo:goals      216 .
dbr:Cristiano_Ronaldo    a              dbo:Athlete .
dbr:Cristiano_Ronaldo    foaf:homepage  "http://cristianoronaldo.com" .
dbr:Rio_Ferdinand        a              dbo:Athlete .
dbr:Rio_Ferdinand        dbp:goals      10 .
```

2. $a_{i(05:02-05:00)} =$

```
dbr:Arvid_Smit     a              dbo:Athlete .
dbr:Barack_Obama   foaf:homepage  "http://www.barackobama.com" .
```

3. $a'_{05:02-05:00} = \emptyset$

Now, we will use the results from Definiton 39 and Definition 40 to compute interesting and potentially interesting changesets.

Definition 41 (Interest Evaluation) *An interest evaluation over a changeset $\Delta(V_{t_1-t_0})$ at time t_1 is a function $e(i_g, \Delta(V_{t_1-t_0}))$ that combines the results from an interest evaluation over deleted triples, $d(i_g, D_{t_1-t_0})$, and an interest evaluation over added triples, $\alpha(i_g, I_{t_1-t_0})$, to return an interesting changeset and potentially interesting changeset as follows:*

$$e(i_g, \Delta(V_{t_1-t_0})) = d(i_g, D_{t_1-t_0}) \quad \chi \quad \alpha(i_g, I_{t_1-t_0}) = \langle \Delta(\tau_{t_1-t_0}), \Delta(\rho_{t_1-t_0}) \rangle$$

where i_g is an interest expression over an evolving dataset, $\Delta(\tau_{t_1-t_0})$ is an interesting changeset (see Definition 42), and $\Delta(\rho_{t_1-t_0})$ is potentially interesting changeset (see Definition 43).

Definition 42 (Interesting Changeset) *Let τ_{t_0} be a target dataset at time t_0. An interesting changeset, $\Delta(\tau_{t_1-t_0})$, for τ_{t_0} at time t_1 is defined as:*

$$\Delta(\tau_{t_1-t_0}) = \left\langle (r_{t_1-t_0} \cup r'_{t_1-t_0}),\ a_{t_1-t_0} \right\rangle$$

where:

- *$r_{t_1-t_0}$ is the set of interesting removed triples, interesting removed optional triples and potentially interesting removed triples with match found in target dataset during candidate generation, $\pi(i_g, D_{t_1-t_0})$,*
- *$r'_{t_1-t_0}$ is the set of triples from target dataset that are related to potentially interesting removed triples computed by $\pi'(i_g, D_{t_1-t_0})$, and*
- *$a_{t_1-t_0}$ is the set of interesting added triples, interesting optional triples and potentially interesting added triples with match found in target dataset during candidate generation, $\pi(i_g, A_{t_1-t_0})$.*

Example 12 *An interesting changeset for our running example is as follows: $\Delta(\tau_{05:02}) = \left\langle (r_{05:02-05:00} \cup r'_{05:02-05:00}),\ a_{05:02-05:00} \right\rangle$*

1. *interesting removed triples – $(r_{05:02-05:00} \cup r'_{05:02-05:00})$:*

```
dbr:Marcel              a              dbo:Athlete .
dbr:Marcel              dbp:goals      1 .
dbr:Cristiano_Ronaldo   dbo:goals      96 .
dbr:Cristiano_Ronaldo   a              dbo:Athlete .
dbr:Cristiano_Ronaldo   foaf:homepage  "http://cristianoronaldo.com" .
```

2. *interesting added triples – $a_{05:02-05:00}$:*

```
dbr:Cristiano_Ronaldo   dbo:goals    216 .
dbr:Cristiano_Ronaldo   a           dbo:Athlete .
dbr:Cristiano_Ronaldo   foaf:homepage  "http://cristianoronaldo.com" .
dbr:Rio_Ferdinand       a              dbo:Athlete .
dbr:Rio_Ferdinand       dbp:goals      10 .
```

Triples that were interesting will be downgraded to potentially interesting and stored in ρ_{t_1}, if deletion involves triples matching at least one triple pattern from interest expression BGP.

Definition 43 (Potentially Interesting Changeset) *Let ρ_{t_0} be a potentially interesting dataset for interest expression i_g at time t_0. A changeset, $\Delta(\rho_{t_1-t_0})$, for ρ_{t_0} at time t_1 is defined as:*

$$\Delta(\rho_{t_1-t_0}) = \left\langle r_{i(t_1-t_0)},\ (a_{i(t_1-t_0)} \cup r'_{t_1-t_0}) \right\rangle$$

where:

- *$r_{i(t_1-t_0)}$ is a set of potentially interesting removed triples,*
- *$a_{i(t_1-t_0)}$ is a set of potentially interesting added triples computed on added triples of a changeset and related triples extracted from target while removing potentially interesting removed triples, and*
- *$r'_{t_1-t_0}$ is the set of triples from target dataset that are related to potentially interesting removed triples computed by $\pi'(i_g, D_{t_1-t_0})$.*

Example 13 *Potentially interesting changeset for our running example is as follows:* $\Delta(\rho_{05:02}) = \left\langle r_{i(05:02-05:00)}, \left(a_{i(05:02-05:00)} \cup r'_{05:02-05:00}\right)\right\rangle$

1. *Potentially interesting removed triples –* $r_{i(05:02-05:00)} = \emptyset$
2. *Potentially interesting added triples –* $\left(a_{i(05:02-05:00)} \cup r'_{05:02-05:00}\right)$

```
dbr:Arvid_Smit     a              dbo:Athlete .
dbr:Barack_Obama   foaf:homepage  "http://www.barackobama.com" .
dbr:Marcel         a              dbo:Athlete .
```

Note: *since all triples in* $r'_{05:02-05:00}$ *are added back to target dataset, they are no longer stored in the potentially interesting dataset.*

Definition 44 (Interesting Update Propagation) *An interesting changeset propagation is an update operation that transforms the target dataset* τ_{t_0} *to the new dataset* τ_{t_1} *and* ρ_{t_0} *to new dataset* ρ_{t_1} *by applying the result of interest evaluation,* $e(i_g, \Delta(V_{t_1-t_0}))$*. That is:*

$$\Upsilon(i_g, \Delta(V_{t_1-t_0})) = \upsilon(\tau_{t_0}, \Delta(\tau_{t_1-t_0})) \ \wedge \ \upsilon(\rho_{t_0}, \Delta(\rho_{t_1-t_0})) = \tau_{t_1} \ \wedge \ \rho_{t_1}$$

- $\Delta(V_{t_1-t_0})$ *is a changeset at time* t_1*,*
- $\upsilon(\tau_{t_0}, \Delta(\tau_{t_1-t_0})) = (\tau_{t_0} \backslash [r_{t_1-t_0} \cup r'_{t_1-t_0}]) \cup a_{t_1-t_0}$ *is changeset propagation of interesting changeset, and*
- $\upsilon(\rho_{t_0}, \Delta(\rho_{t_1-t_0})) = (\rho_{t_0} \backslash r_{i(t_1-t_0)}) \cup (a_{i(t_1-t_0)} \cup r'_{t_1-t_0})$ *is changeset propagation of potentially interesting changeset.*

Example 14 *Propagation of an interesting changeset of Example 12 to the target dataset,* τ_{t_0} *and potentially interesting changeset of Example 13 to the potentially interesting dataset*ρ_{t_0} *transforms the datasets to:*

```
dbr:Cristiano_Ronaldo  dbo:goals 216 .
dbr:Cristiano_Ronaldo  a dbo:Athlete .
dbr:Cristiano_Ronaldo  foaf:homepage
    "http://cristianoronaldo.com" .
dbr:Rio_Ferdinand  a          dbo:Athlete .
dbr:Rio_Ferdinand  dbp:goals        10 .
```

Listing 8.3: Resulting target dataset

```
dbr:Arvid_Smit     a     dbo:Athlete .
dbr:Barack_Obama   foaf:homepage
      "http://www.barackobama.com" .
dbr:Marcel      a        dbo:Athlete .
```

Listing 8.4: Potentially interesting dataset after change propagation

8.4 Managing Co-evolution

Figure8.4 illustrates the co-evolution between two RDF datasets. Initially, a slice of source dataset is used to create a target dataset, i.e., the target dataset T_{t_0} is sliced from the source dataset S_{t_0} of dataset S at time t_0. Both the source and target datasets evolve themselves with the passage of time, e.g., these datasets evolve to S_{t_j} and T_{t_j} during timeframe $t_i - t_j$, while $t_i < t_j$. Changes from S_{t_j}, denoted by $\Delta(S_{t_j-t_i})$, are propagated to the target and vice versa by the RDF data synchronization component. For synchronization, changes from both source and target datasets are compared to identify conflicts. The resolved conflicts are applied on the source and target datasets to vanish inconsistencies, for example,

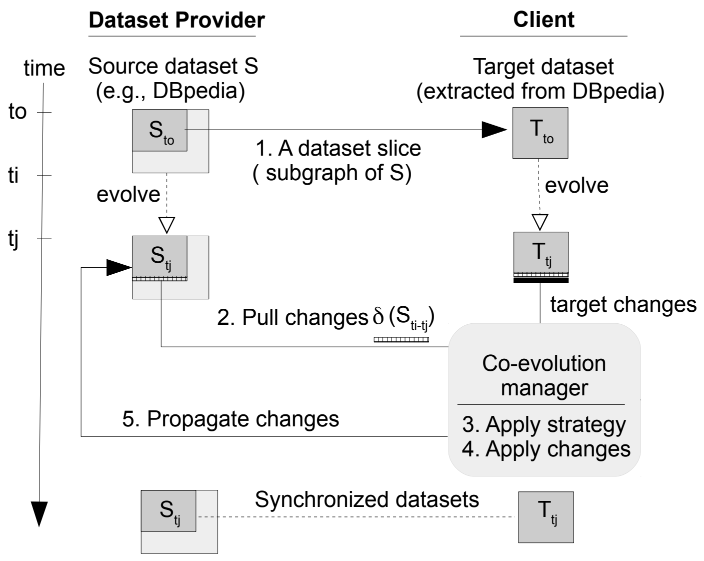


Figure 8.4: Co-evolution of linked datasets

at time point t_j, the co-evolution manager identifies the conflicts and resolves them. The conflicts are resolved and final changes are merged in both datasets.

8.4.1 Conflict

Our co-evolution strategy aims at dealing with changesets from either the source or target dataset and provide a suitable reconciliation strategy. Various strategies can be employed for synchronizing datasets. When we synchronize the target T_{t_i} with source S_{t_i}, there may exist triples which have been changed in both datasets. These changed triples may be conflicting.

Definition 45 (Potential Conflict) *Let us assume that a synchronization is required for a given time slot $t_i - t_j$. $\Delta(S_{t_j - t_i})$ is the changeset of the source dataset and $\Delta(T_{t_j - t_i})$ is the changeset of the target dataset. A potential conflict is observed when there are triples $x_1 = (s, p, o_1) \in S_{t_j} \wedge x_2 = (s, p, o_2) \in \Delta(T_{t_j - t_i}) \wedge x_2 \notin S_{t_j} = S_{t_i} \cup \Delta(S_{t_j - t_i})$ with $o_1 \not\equiv o_2$.*

Taking $o_1 \not\equiv o_2$ as an indication for a conflict is subjective; in the sense that the characteristics of the involved property p influences the decision. Consider two triples (s, p, o_1) and (s, p, o_2). If p is a functional data type property, two triples are conflicting iff the object values o_1 and o_2 are not equal. However, if the property p is a functional object property, these two triples are conflicting if the objects are or can be inferred to be different (e.g. via `owl:differentFrom`). Another property which needs special consideration is `rdf:type`. For this property it is necessary to check whether o_1 and o_2 belong to disjoint classes. Only then these triples would be conflicting. For example, `s1 rdf:type Person` and `s1 rdf:type Athlete` are not conflicting if `Athlete` is a subclass of `Person` (i.e. not disjoint). Thus, the process of detecting conflicts is considering the inherent characteristics of the involved property.

8.4.2 Synchronization Strategies

In the following, we list possible strategies for synchronization. We consider the time frame $t_i - t_j$, where in the time t_i, the source and target datasets are synchronised and until time t_j, both source and target datasets have been evolving independently. Before applying synchronization, the state of the source dataset is $S_{t_j} = S_{t_i} \cup \Delta(S_{t_j - t_i})$ and the target dataset is $T_{t_j} = T_{t_i} \cup \Delta(T_{t_j - t_i})$.

Strategy I: This synchronization strategy prefers the source dataset and ignores all local changes on the target dataset; thus, the following requirement is necessary. Therefore, the target dataset ignores all triples $\{x \mid x \notin \Delta(S_{t_j - t_i}) \ \wedge \ x \in \Delta(T_{t_j - t_i})\}$ and adds only the triples $\{y \mid y \in \Delta(S_{t_j - t_i})\}$. After synchronization, the state of source dataset is $S_{t_j} = S_{t_i} \cup \Delta(S_{t_j - t_i})$ and the state of the target dataset is $T_{t_j} = T_{t_i} \cup \Delta(S_{t_j - t_i})$. Thus, the inclusion requirement is met and $T_{t_j} \subseteq S_{t_j}$. A special case of this strategy is when the target is not evolving.

Strategy II: With this strategy, the target dataset is not synchronized with the source dataset and keeps all its local changes. Thus, the target dataset is not influenced by any change from the source dataset and evolves locally. After synchronization, at time t_j, the state of the target dataset is $T_{t_j} = T_{t_i} \cup \Delta(T_{t_j - t_i})$, and the state of the source dataset is $S_{t_j} = S_{t_i} \cup \Delta(S_{t_j - t_i})$. It allows for synchronized replicas only if data is deleted. There is no synchronization if triples in the target dataset are updated or new triples are included.

Strategy III: This synchronization strategy respects the changesets of both source and target datasets except that it ignores conflicting triples. Here, the set of triples in which conflicts occur is $X = \{x_1 = (s, p, o_1) \in S_{t_j} \wedge x_2 = (s, p, o_2) \in \Delta(T_{t_j - t_i}) \wedge x_2 \notin S_{t_j}$ with $o_1 \not\equiv o_2\}$[9]. With Strategy III, the set of conflicting triples X is removed from the target dataset while the source changeset $\Delta(S_{t_j - t_i})$ and the target changeset $\Delta(T_{t_j - t_i})$ are added. After synchronization, the state of the source dataset is $S_{t_j} = (S_{t_i} \cup \Delta(S_{t_j - t_i}) \cup \Delta(T_{t_j - t_i}))\backslash X$ and the state of the target dataset is $T_{t_j} = (T_{t_i} \cup \Delta(T_{t_j - t_i}) \cup \Delta(S_{t_j - t_i}))\backslash X$. Thus, the inclusion requirement is met.

Strategy IV: This synchronization strategy also respects the changesets of both source and target datasets. In addition, it includes conflicting triples after resolving the conflicts. Here, we consider the set of triples in which conflict occurs as $X = \{x_1 = (s, p, o_1) \in S_{t_j} \wedge x_2 = (s, p, o_2) \in \Delta(T_{t_j - t_i}) \wedge x_2 \notin S_{t_j}$ with $o_1 \not\equiv o_2\}$. The conflicts over these triples should be resolved. It can be resolved using some resolution policy as described in [87]. Table 8.1 shows a list of various policies for resolving the conflicts. Conflict resolution results in a new set of triples called Y whose triples are originated from X but their conflicts have been resolved. Then, this new set (i.e. Y) is added to the both source and target datasets. After synchronization, the state of the source dataset is $S_{t_j} = ((S_{t_i} \cup \Delta(S_{t_j - t_i}) \cup \Delta(T_{t_j - t_i}))\backslash X) \cup Y$ and the state of target dataset is $T_{t_j} = ((T_{t_i} \cup \Delta(T_{t_j - t_i}) \cup \Delta(S_{t_j - t_i}))\backslash X) \cup Y$. Thus, the inclusion requirement is met.

8.4.3 Co-evolution Approach

Our approach allows a user to choose a synchronization strategy defined in Section 8.4.2. Below, we describe the status of the source and target datasets after applying each synchronization strategy.

We define a function *CDR* (*Conflict Detection and Resolution*), which (i) identify conflicts for the case of strategy III and strategy IV, and then (ii) resolve conflicts only in case of strategy IV. Our approach considers triple-based operations, explained below using seven cases, to identify conflicts. Consider three triples $x_1 = (s, p, o_1)$, $x_2 = (s, p, o_2)$, and $x_3 = (s, p, o_3)$ which are in conflict with each other $x_1 \in \Delta(S_{t_j - t_i}) \wedge x_2 \in \Delta(T_{t_j - t_i}) \wedge x_3 \in \{\Delta(S_{t_j - t_i}) \wedge \Delta(T_{t_j - t_i})\} \wedge o_1 \not\equiv o_2 \not\equiv o_3$. In the following we present seven cases of evolution causing conflicts. For the first three cases (I-III), the conflict resolution is straightforward. But for the cases IV-VII, we have to employ a conflict resolution policy to decide about triples x_1 and x_2 (D^S and A^S refers to the deleted and added triples from source dataset, respectively. Similarly, D^T and A^T refers to the deleted and added triples from target dataset):

[9] Set of conflicting triples selected after considering the inherent characteristics of the involved property. In rest of the chapter, we say potential conflict a conflict, unless otherwise specified.

Category	Policy	Function	Type	Description
Deciding	Roll the dice	Any	A	Pick random value.
	Reputation	Best Source	A	Select the value from the preferred dataset.
	Cry with the wolves	Global vote	A	Select the frequently occurring value for the respective attribute among all entities.
	Keep up-to-date	First*	A	Select the first value in order.
		Latest*	A	Select the most recent value.
	Filter	Threshold*	A	Select the value with a quality score higher than a given threshold.
		Best*	A	Select the value with highest quality score.
		TopN*	A	Select the N best values.
Mediating	Meet in the Middle	Standard deviation, variance	N	Apply the corresponding function to get value.
		Average, median	N	Apply the corresponding function to get value.
		Sum	N	Select the sum of all values as the resultant.
Conflict Ignorance	Pass It On	Concatenation	A	Concatenate all the values to get the resultant.
Conflict Avoidance	Take the Information	Longest	S, C, T	Select the longest (non-NULL) value.
		Shortest	S, C, T	Select the shortest (non-NULL) value.
		Max	N	Select the maximum value from all.
		Min	N	Select the minimum value from all.
	Trust Your Friends	Choose Depending*	A	Select the value that belongs to a triple having a specific given value for another given attribute.
		Choose Corresponding	A	Select the value that belongs to a triple whose value is already chosen for another given attribute.
		Most Complete*	A	Select the value from the dataset (source or target) that has fewest NULLs across all entities for the respective attribute.

* - requires metadata

Table 8.1: Conflict resolution policies and functions: A - All, S - String, C - Category (i.e., domain values have no order), T - Taxonomy (i.e., domain values have semi-order), N - Numeric.

- ***Case I:*** x_1 is added to T_{t_j} if x_1 is added by the source dataset and x_2 is deleted from the target dataset: $x_1 \in \Delta(A^S_{t_j - t_i}) \land x_2 \in \Delta(D^T_{t_j - t_i})$.
- ***Case II:*** x_1 is added to T_{t_j} if x_1 is modified by the source dataset and x_2 is deleted from the target dataset: $x_1 \in \Delta(A^S_{t_j - t_i}) \land x_2 \in \Delta(D^S_{t_j - t_i}) \land x_2 \in \Delta(D^T_{t_j - t_i})$.
- ***Case III:*** x_2 is added to S_{t_j} if x_1 is deleted from the source dataset and x_2 is modified in the target dataset: $x_1 \in \Delta(D^S_{t_j - t_i}) \land x_2 \in \Delta(A^T_{t_j - t_i}) \land x_1 \in \Delta(D^T_{t_j - t_i})$.
- ***Case IV:*** if the triple x_1 is added to the source dataset and x_2 is added to the target dataset: $x_1 \in \Delta(A^S_{t_j - t_i}) \lor x_2 \in \Delta(A^T_{t_j - t_i})$.
- ***Case V:*** if x_3 is modified by both source and target datasets: $x_2 \in \Delta(A^S_{t_j - t_i})^+ \land x_3 \in \Delta(D^S_{t_j - t_i}) \land x_1 \in \Delta(A^T_{t_j - t_i}) \land x_3 \in \Delta(D^T_{t_j - t_i})$.
- ***Case VI:*** if x_1 is modified by the target dataset: $x_1 \in \Delta(A^S_{t_j - t_i}) \land x_2 \in \Delta(A^T_{t_j - t_i}) \land x_1 \in \Delta(D^T_{t_j - t_i})$.
- ***Case VII:*** if x_1 is modified by the source dataset: $x_2 \in \Delta(A^S_{t_j - t_i}) \land x_1 \in \Delta(D^S_{t_j - t_i}) \land x_1 \in \Delta(A^T_{t_j - t_i})$.

As we discussed earlier, whether a conflict between two triple exists depends heavily on the type of property. Consider two triples (s, p, o_1) and (s, p, o_2), if p is `rdfs:label`, we measure the similarity between o_1 and o_2 using the Levenshtein distance. We pick both values of `rdfs:label` if their similarity is below a certain threshold otherwise we treat them as conflicting.

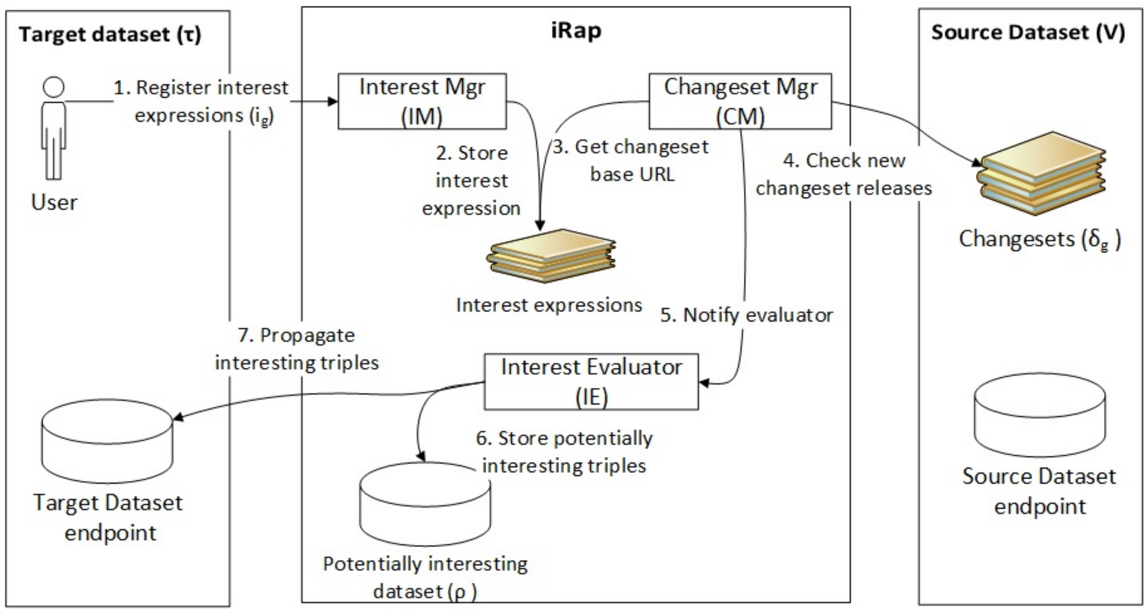

Target dataset (τ)
iRap
Source Dataset (V)
User
1. Register interest expressions (i_g)
Interest Mgr (IM)
Changeset Mgr (CM)
2. Store interest expression
3. Get changeset base URL
4. Check new changeset releases
Changesets (δ_g)
Interest expressions
5. Notify evaluator
7. Propagate interesting triples
Interest Evaluator (IE)
6. Store potentially interesting triples
Target Dataset endpoint
Source Dataset endpoint
Potentially interesting dataset (ρ)

Figure 8.5: Architecture of the iRap interest-based RDF update propagation framework.

8.5 iRap: RDF Update Propagation Framework

In this section we describe the architecture of our interest-based update propagation framework, iRap, and its implementation. iRap is implemented in Java using Jena-ARQ. It is available as open-source[10] and consists of three modules: (1) *Interest Manager* (IM), (2) *Changeset Manager* (CM) and (3) *Interest Evaluator* (IE), each of which can be extended to accommodate new or improved functionality.

Changeset evaluation starts after a user registers an interest expression using the IM service, as shown in Figure 8.5. The CM module fetches a list of changeset folders from interest expressions and regularly (configurable) checks for new changesets. After downloading and decompressing new changesets, the CM notifies the IE, which then imports a list of interest expressions registered for this particular changeset through the IM and initiates the evaluation. Resulting interesting triples are propagated to the target dataset whereas potentially interesting triples are stored in the potentially interesting dataset (ρ).

8.6 Empirical Evaluation

In this section, we empirically evaluate our proposed approach in two parts. First, we evaluate the interest-based update propagation approach to confirm the validity and performance of the iRap framework in propagation of updates where the target dataset is not allowed to evolve over time. In the second part of the evaluation, we assess the synchronization and conflict resolution strategies. We also study the effect of conflict identification and resolution strategies on the quality of the datasets using different quality metrics.

8.6.1 Evaluating iRap Update Propagation

To evaluate the interest-based update propagation approach, we performed experiments on the iRap framework using changesets published by DBpedia and compared the results with the DBpedia Live

[10] `https://github.com/EIS-Bonn/iRap`

Date	Oct 01	Oct 02	Oct 03	Oct 04-12	Oct 13	Oct 14	Oct 15
Total Changesets	0	1,621	1,755	0	5,352	751	2,578

Table 8.2: Distribution of DBpedia Live changesets published October 01-15, 2014.

```
CONSTRUCT WHERE {
   ?footballer   a   dbo:SoccerPlayer  .
   ?footballer   foaf:name   ?name.
   ?footballer   dbo:team    ?team  .
   ?team         rdfs:label  ?teamName.
}
```

Listing 8.5: I_1 – Football interest query

```
CONSTRUCT WHERE {
   ?location   a              ?type  .
   ?location   wgs:long       ?long  .
   ?location   wgs:lat        ?lat  .
   ?location   rdfs:label     ?label  .
   ?location   dbo:abstract   ?abstract.
   OPTIONAL {
   ?location  dcterms:subject  ?subject
        }
}
```

Listing 8.6: I_2 – Location interest query

Mirror tool. The comparison considers two cases: using iRap to update a previously-established local replica of i) an entire remote dataset ii) a subset of a remote dataset. These two cases simulate two ways in which iRap can be used: i) using interest-based changeset propagation for future updates of a local copy of a large dataset or ii) starting with a new subset of the large dataset. Both cases will only consider strategy I, in Section 8.4.2.

Experimental Setting

In order to test our approach we used the DBpedia dump[11] of September 30, 2014 for the initial setup of the target datasets for two different application domains, namely, *Location and Football* datasets. Changesets published between October 01 and October 15, 2014 were used for evaluation (see Table 8.2). Changesets are not sequential with modified date but with extraction from DBpedia Live, as discussed in the DBpedia mailing list. Initially we set up two Jena TDB datasets for each target dataset from the DBpedia dump. We loaded all triples from the dump to the Location dataset, whereas for the Football dataset we only loaded a slice corresponding to interesting triples matching Listing 8.5. Initially, the Location dataset contains all triples from DBpedia yielding a total of 3 billion triples, whereas the Football dataset contains only 265,622 triples. A total of 12,057 changesets (pairs of removed and added .nt.gz files) have been published in the evaluation timeframe. The evaluation comprises two interest expressions, I_1 and I_2. I_1 comprises a non-disjoint BGP containing 4 triple patterns with a maximum of two variables per triple pattern (object-subject join), Listing 8.5. I_2 comprises a non-disjoint BGP containing 5 triple patterns with a maximum of two variables per triple pattern (subject-subject joins) and one an OGP containing one triple pattern, Listing 8.6.

We set up two target datasets and potentially interesting dataset using Jena TDB and jena-fuseki for each dataset. The potentially interesting dataset stores potentially interesting triples for each interest expression within a named graph. All experiments were carried out on a 64-bit machine with Windows 7, Intel(R) Core i7-4770 CPU, 16GB RAM and 1TB HD.

[11] `http://live.dbpedia.org/dumps/dbpedia_2014_09_30_00_00.fixed.ttl.gz`

Day	Total Removed	Interesting Removed	Total Added	Interesting Added	Potentially Interesting	Elapsed (in minutes)
1	1,895,179	9,065	2,051,976	184	169,554	15.18
2	1,748,511	4,865	2,384,232	155	168,856	20.85
3	1,716	0	10,728,855	45,429	684,491	69.86
4	449	0	1,522,939	7,970	97,300	10.17
5	1,677	0	5,234,788	19,598	333,232	60.06

Table 8.3: Comparison of results for Football App

Day	Total Removed	Interesting Removed	Total Added	Interesting Added	Potentially Interesting	Elapsed (in minutes)
1	1,895,179	77,377	2,051,976	7,093	430376	166.59
2	1,748,511	82,461	2,384,232	7,301	509,972	242.62
3	1,716	0	10,728,855	259,587	2,002,271	417.87
4	449	0	1,522,939	27,292	280,718	64.41
5	1,677	0	5,234,788	100,073	972,284	176.78

Table 8.4: Comparison of results for Location App

Results and Discussion

Figure 8.6 and Figure 8.7 summarizes our experimental results for two target datasets shows the growth of the potentially interesting dataset. Results of the interest evaluation for the Football dataset are presented in Table 8.3. From the overall changesets considered for this evaluation, in Table 8.2, only 0.38% of the removed and 0.335% of the added triples were identified as interesting for the Football dataset. The average changeset publication interval was 18.81s and average time required for a changeset evaluation is 0.87s. This shows that iRap efficiently performs changeset propagations way before the next changeset is published.

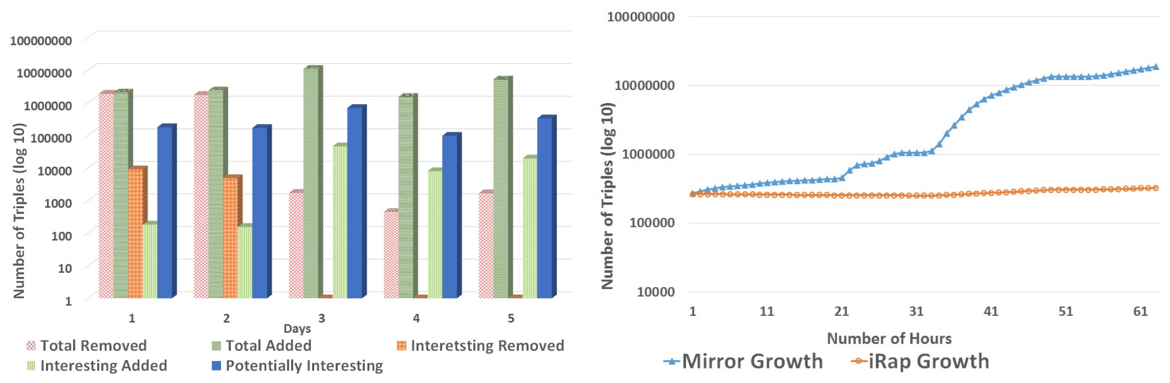
Results of the interest evaluation for the Location dataset are shown in Table 8.4. From the overall changesets considered for this evaluation, in Table 8.2, only 4.38% of the removed and 1.81% of the added triples were interesting for the Location dataset. The average time spent for a changeset evaluation is 5.31s. The interest evaluation for the Location dataset takes longer than Football dataset, because of the number of triples in the target dataset was the full DBpedia. Figure 8.6a shows the number of triples published per day and the number of interesting triples and potentially interesting triples found from interest evaluation for Football dataset. Figure 8.6b shows the dataset growth comparison between iRap and a full mirror approach. As the figure clearly shows, iRap managed datasets are almost two orders of magnitude smaller and grow much slower than with a mirror approach. Note that the growth for each datasets is calculated by subtracting the number of removed triples from and adding the number of added triples to the total number of triples in the dataset.

We observed a logarithmic growth of the potentially interesting dataset for Location and Football datasets. This is due to the number of variables used in triple patterns, and the number and type of triple patterns in interest expression. For example, the Football dataset interest query contains the common predicates `foaf:name` and `rdfs:label` which are used in almost all resources and thus result in many potentially interesting triples. Again, the average processing time per changeset is always way below the average time between two changesets. The correctness of the resulting triples from the first changesets, for Football dataset interest expression, was checked by manual inspection.

8.6.2 Evaluating Co-evolution Strategies

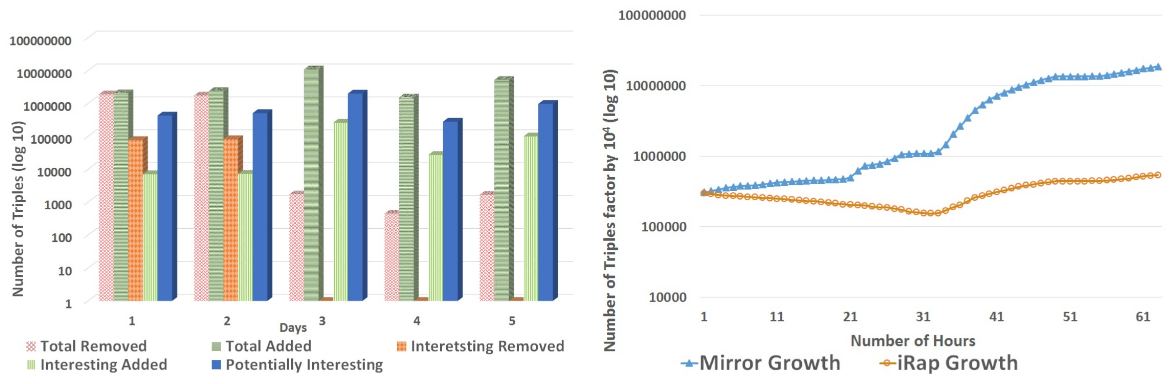
In order to assess the co-evolution approach for synchronization and conflict identification/resolution, we prepare a testbed based on a slice of DBpedia using the following Interest Expression, I, query:

```
CONSTRUCT WHERE  {
```



(a) Changes per day (b) Dataset growth

Figure 8.6: Evaluation results for Football dataset.



(a) Changes per day (b) Dataset growth

Figure 8.7: Evaluation results for Location dataset.

```
        ?s        a                 Politician ;
                  foaf:name         ?name ;
                  dbo:nationality   ?nationality ;
                  dbo:abstract      ?abstract ;
                  dbp:party         ?party ;
                  dbp:office        ?office
        OPTIONAL { ?s  foaf:depiction  ?depiction }
}
```

The extracted dataset is used as the initial source and target dataset. Then, we collect a series of changesets from DBpedia-live published from September 01, 2015 to October 31, 2015 using iRap [106]. We found a total of 304 changesets. These changesets are leveraged to simulate updates of the source and target datasets. We randomly select a total of 91 addition parts of changesets and altered values of their triples. Table 8.5 provides the number of triples of initial target, source and their associated changesets before synchronization. Initially, we have *200082* triples with *163114* unique objects in T_{t_i} where $t_i = September01, 2015$.

S_{t_i}	T_{t_i}	$\Delta(S_{t_j-t_i})^+$	$\Delta(S_{t_j-t_i})^-$	$\Delta(T_{t_j-t_i})^+$	$\Delta(T_{t_j-t_i})^-$
200082	200082	948	160	11725	81

Table 8.5: Number of triples in the source, target, and changesets for a given time frame

Results and Discussion

Given a timeframe $t_i - t_j$ [12], the goal is to synchronize source and target datasets. To do that, we define five different scenarios. In four scenarios, we apply subsequently the strategy (I-IV) over all predicates of the changesets and measure the performance. For the last scenario, we apply two strategies in a combined form on the changesets where we select strategy IV for predicate *dbp:office*, and strategy I for predicates *dbp:party*, *dbo:nationality*, *rdf:type*, *foaf:name*, *dbo:abstract*, and *foaf:depiction*. For all predicates using strategy IV, we select the resolution function 'any'. Table 8.6 provides the number of triples produced as a result of synchronizing S_{t_i} and T_{t_i} in each scenario. The updated changesets are sent back to the source and target for synchronization purpose. The number of conflicting triples found in scenarios 3, 4, and 5 are shown in Table8.6.

Scenario	$\Delta(S_{t_j-t_i})^+$	$\Delta(S_{t_j-t_i})^-$	$\Delta(T_{t_j-t_i})^+$	$\Delta(T_{t_j-t_i})^-$	Conflicting triples	RunTime (seconds)
1	0	0	948	160	-	0.0
2	0	0	11725	81	-	0.0
3	11682	81	12060	81	343	0.5
4	11800	195	12186	81	343	2.0
5	5227	131	6081	121	186	0.2

Table 8.6: Results of synchronization

The running time of the five different scenarios is also shown in Table 8.6 (These times are recorded only for the execution of synchronization part and do not include data loading time). Evaluation showed that strategy IV (performed in scenario IV) needs more time even from strategy III (performed in scenario III) where all conflicts were detected but not resolved.

Synchronization influences data quality specially in terms of data consistency. To evaluate the usefulness of the synchronization approach, we use three data quality metrics i.e. (1) *completeness*, (2) *conciseness*, and (3) *consistency* described as follows:

1. Completeness refers to the degree to which all required information is present in a dataset [111]. We measure it for source and target changesets to identify which helps more in completeness. We measure it using

$$\frac{Number\,of\,unique\,triples\,in\,synchronised\,dataset}{Number\,of\,unique\,triples\,in\,(initial\,dataset\ \cup\ changeset)}$$

2. Consistency states that the values should not be conflicting. We measure it using

$$\frac{Number\,of\,non\text{-}conflicting\,triples\,in\,synchronized\,dataset}{Number\,of\,triples\,in\,(initial\,dataset\ \cup\ source\,and\,target\,changesets)}$$

3. Conciseness measures the degree to which the dataset does not contain redundant information using

$$\frac{Number\,of\,unique\,triples\,in\,dataset}{Number\,of\,all\,triples\,in\,dataset}$$

[12] 09/01/2015-10/31/2015.

Conciseness (before synchronization) is computed using initial target dataset and source and target changesets. We compute these metrics for all the assumed scenarios, the results are shown in Table 8.7. For our sample case study, we found almost equal contribution of both source and target changesets in reducing the missing information. However, we found minimum *163,191* number of unique objects using strategy II and maximum *163,591* number of unique objects using strategy IV. Note that strategy 1 and strategy II may not necessarily increase the number of unique triples as they do not consider about conflicts. It can be observed by analyzing the scenario 1 where the role of source changesets in completeness is 99% which is less than the target contribution. Through evaluation, we found significant increase in conciseness for all strategies.

Scenario	**Completeness (source)**	**Completeness (target)**	**Consistency**	**Conciseness (before synchronization)**	**Conciseness (after synchronization)**
1	99%	100%	-	77%	81%
2	99%	99%	-	77%	81%
3	99%	100%	94%	77%	81%
4	99%	100%	94%	77%	81%
5	99%	100%	-	77%	81%

Table 8.7: Synchronization effect on completeness, consistency, and conciseness

8.7 Summary

In this chapter, we present a novel approach for interest-based RDF update propagation that consistently maintains a full or partial replication of large LOD datasets and to deal with co-evolution, which refers to mutual propagation of the changes between a replica and its origin dataset. We demonstrate the validity of the approach through detailed formalization and their application in a reference implementation of the iRap Framework. Furthermore, we demonstrate the approach using formal definitions of all the concepts required for realizing co-evolution of RDF datasets and implemented it using different strategies. Using the co-evolution process, we address synchronization and conflict resolution issues. A thorough evaluation of the approach is performed using large-scale real-world data dumps and changesets regularly provided by a renowned LOD dataset. The results of the evaluation indicates that our method can significantly cut down on both the size of the data updates required to consistently maintain a localized dataset replication up-to-date, as well as the speed by which such updates can take place. We evaluate the co-evolution approach using data quality metrics completeness, conciseness, and consistency. The results of the evaluation indicates that our method can significantly improve the quality of dataset.

CHAPTER 9

Conclusions and Future Directions

In this thesis, we study the problem of federated query processing over autonomous, distributed, and heterogeneous data sources in a Semantic Data Lake. We also study the problem of data update propagation and co-evolution of datasets when data need to be synchronized between the original and target sources. Different techniques and frameworks that can help transform Big Data to actionable knowledge are presented. In particular, we propose a model to describe data sources that are member of the federation, i.e., part of the Semantic Data Lake, as well as techniques to express update interests and strategies for conflict resolution between co-evolving data sources. Furthermore, we propose techniques for data source selection, query decomposition, and execution plan generation considering the variety dimension of the underlying data sources as well as their privacy and access control policies.

9.1 Revisiting the Research Questions

RQ1: How can we describe the semantics encoded in heterogeneous data sources?

Chapter 4 present the proposed data source description model, i.e., RDF Molecule Template (RDF-MT), for autonomous, distributed, and heterogeneous data sources. RDF-MTs are an abstract description of entities in the Semantic Data Lake that represent the same semantic concept. RDF-MTs also describe implementation of different parts of the RDF molecules that is stored in each data sources in the federation. We present a high-level analysis of different state-of-the-art benchmarks using RDF-MTs and provide a birds-eye view of different characteristics of each federation. We evaluate the performance of a query engine using RDF-MT based source description compared to descriptions generated using other graph partitioning techniques. The results show that RDF-MT based source descriptions improve the performance of the federated query engine in terms of total execution time as well as answer completeness. Therefore, we can conclude that RDF-MT based source descriptions capture the semantics encoded in heterogeneous data sources and improved the performance of federated query processing.

RQ2: How can features represent in data source descriptions be employed to guide the query processing over heterogeneous data sources?

To answer this research question, we employ the proposed data source description model, i.e., RDF-MTs, and devise techniques for query decomposition, source selection, and planning and optimization. First, in Chapter 5, we present the source selection and query decomposition technique, the MULDER approach, that exploit the RDF Molecule Template based source description model. We evaluate the MULDER approach over three different benchmarks for evaluating federated SPARQL query engine, thus, focusing on the source selection and query decomposition sub-problem of federated query processing. MULDER also employ RDF-MTs for query planning and optimization over RDF data sources in the federation. We compare the MULDER federated query engine against state-of-the-art federated SPARQL query engines; the observed results show that MULDER is able to minimize execution time and maximize answer completeness compared to the state-of-the-art federated query engines. Then, in Chapter 6, we present the query planning and optimization technique, the Ontario approach, in the presence of heterogeneous data sources in the federation. The Ontario approach exploits the RDF-MT based source description model to guide the optimization of execution plan. Ontario employs a heuristic based optimizer that considers the heterogeneity of data sources as described by the RDF-MTs and the characteristics of the star-shaped groups to find an efficient execution plan. The evaluation results show that even though, data sources behave differently on different characteristics of star-shaped groups, Ontario is able to create efficient and effective plans where physical operators are selected correctly. Both MULDER, as federated query engine over homogeneous data sources, and Ontario, as a federated query engine over heterogeneous data sources, are able to exploit the features represented in RDF-MT based data source descriptions to maximize answer completeness and minimize execution time. Hence, we can conclude that the proposed techniques outperform state-of-the-art techniques by employing the RDF-MT based source descriptions to guide the query processing against distributed, autonomous, and heterogeneous data sources in a Semantic Data Lake.

RQ3: How can privacy and access control requirements be encoded in data source descriptions and be used for enforcement during query processing over federation of data sources?

Chapter 7 present the privacy-aware data source descriptions and federated query processing techniques. The privacy and access control policies of each data sources in a federation are encoded in a privacy-aware RDF-MTs; i.e., RDF-MTs are able to encode privacy and access control requirements. We present a privacy-aware federated query processing engine, the BOUNCER approach, that is able to employ the privacy-aware RDF-MTs during source selection, query decomposition, and execution plan generation. We empirically evaluate the effectiveness and efficiency of BOUNCER; the results showed that BOUNCER is able to minimize execution time as well as maximize answer completeness by selecting the data sources correctly and generating valid query plans that enforce the privacy and access control requirements of the data sources. We show that the RDF-MT based source descriptions can encode the privacy and access control requirements set by data providers at different level of granularity. Privacy-aware RDF-MT based source description enables a privacy-aware federated query processing technique to select relevant data sources as well as to generate a valid execution plan that can provide results to answer the given query without violating the such requirements within a reasonable time.

RQ4: How can data source descriptions be used for propagation of updates for managing (co)evolution of data sources?

In Chapter 8, we present the interest-based update propagation and co-evolution approach. We propose a graph-pattern based technique to define update interests and formalize the problem of interest-based update propagation. We show that RDF-MTs can encode such interest expression for updates, hence, called evolving RDF-MTs. We implement an update propagation framework, iRap, based on our formal definition of the problem of update propagation that propagates updates from the original data source to its replicas by filtering on interesting parts of updates. When mutual propagation of updates between original and replica data sources is allowed, i.e., co-evolution of data sources, different conflicts may arise. We propose different synchronization strategies to manage co-evolution of data source and employ different types of conflict resolution policies and functions. The evaluation results show that the interest-based update propagation approach reduced the amount of data required to be transferred and handled by application in several order of magnitude. Furthermore, synchronization and conflict resolution techniques are able to improve the quality of data sources evaluated in three data quality metrics; completeness, conciseness, and consistency. We also show that RDF-MT based source descriptions can encode update interest expression to describe interest in propagating updates for evolving RDF-MTs. We argue that the proposed technique enables applications to access only interesting and up-to-date data stored locally, in their premises, and be able to co-evolve with the original data source without compromising data quality.

9.2 Open Issues and Future Directions

In this section, we describe open issues and future direction of this work. The first open challenge to be tackled is the creation of mappings from raw data to ontology concepts. Though creating such mappings manually leads to higher accuracy, it could become unmanageable when the number of documents grows. Machine learning based techniques can be used to aid this process by suggesting concepts to label and filter possible equivalences between data points. Furthermore, our source description approach performs no further processing on RDF-MTs when equivalent semantic concepts are expressed in different ontologies. We are aware that there are some existing approaches with regard to ontology alignment. Another open challenge lies on privacy and access control enforcement over a set of queries. As some pieces of data can be derived by combing other data points, our approach is limited to only single query request, i.e., only privacy and access policies for a given query is enforced. We consider this as a limitation of scope which can further be extended to consider multiple requests and definition of policies over combination of predicates. We envision the following future works:

- **Replication-aware federated query processing.** Extend the techniques to be replication aware when selection data sources and generating efficient query execution plan. The replications can be at level of RDF-MTs or data source as a whole. RDF molecules can be replicated across data sources in the Semantic Data Lake. In such cases, the federated query processing technique should be able to utilize this during source selection, decomposition and planning.

- **Selective knowledge graph materialization.** As the increase in number of heterogeneous data sources and their volume could negatively affect the performance of query processing, materialization of the knowledge graph can be an option for applications that require high latency. The Ontario approach can be extended to generate a materialized knowledge graph on-demand from a set of RDF-MTs in the data source description. RDF-MTs can encode required predicates and data sources to materialize the knowledge graph.

- **Semantic labeling for mapping generation.** As the data sources in the Semantic Data Lake comprises heterogeneous data sources, semantic labeling of entities and predicates is very important to make data sources interoperable. In this thesis, we assume such labeling is already defined and represented by a rule-based mappings. This could become cumbersome, when the number of data sources are large. Semantic Data Lakes could contains thousands to hundreds of thousands of documents, data files, and tables, automatic labeling with human in the loop becomes the way to ease the semantic labeling creation process.

- **Efficient wrappers for heterogeneous sources.** Especially for big volume data sources, wrappers are the main bottlenecks for federated query processing engines. Specifically for data sources that have inefficient local execution engines and insufficient resources for processing queries, efficient wrappers can play an important role in minimizing the overall execution time. They could be adaptive to the conditions of the sources and communicate the status with the federated query engine, so that the execution plan can exploit this information and adopt its plan accordingly.
- **Efficient RDF-MT creation and updates for frequently evolving data sources.** RDF-MT based source descriptions should be maintained and up-to-date when data sources change. Efficient RDF-MT creation and maintenance is important.
- **Question answering over federation of data sources.** The RDF-MT based source descriptions can be exploited for query translation over a federation of data sources. Current state-of-the-art query generation techniques for question answering employs a walk over a centralized knowledge graph [112]. Albeit effective, these techniques suffer from high latency of the query generation and execution. The RDF-MT based source descriptions can be extended to encode information that can be exploited for generating a query which can then be executed over a federation of data sources.
- **Extend the Ontario approach to support streaming data sources.** Data in motion can be part of the federation of data sources. In this thesis, we address some of the challenges in transforming Big Data to actionable knowledge mainly focusing on variety and volume. Another interesting direction for the Ontario approach to address in the future would be the velocity dimension, data in motion, of Big Data.

9.3 Closing Remarks

The growing amount of data being available on the Web demands efficient data integration approaches able to transform Big Data into actionable knowledge on which decisions can be made. Transforming Big Data into actionable knowledge demands novel and scalable techniques for enabling not only Big Data ingestion and curation, but also for efficient large-scale semantic data integration, exploration, and discovery. In this thesis, we have shown the proposed semantic description of data sources positively impacted federated query processing techniques, and allows for improving both answer completeness and query execution time. Moreover, we showed that RDF-Molecule Templates based source descriptions allows to define a higher granularity of privacy and access control policies over sensitive data and for describing update interest over an evolving datasets. Finally, the approaches presented in this thesis and the pieces of software produced are being applied to different application domains and taking part in European research projects, e.g., WDAqua[1], iASiS[2], QualiChain[3], and BigDataEurope[4].

[1] http://wdaqua.eu
[2] http://project-iasis.eu
[3] http://qualichain-project.eu
[4] https://www.big-data-europe.eu

Bibliography

[1] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen and S. Belfkih, *Big Data technologies: A survey*, Journal of King Saud University - Computer and Information Sciences **30** (2018) 431, ISSN: 1319-1578, URL: `http://www.sciencedirect.com/science/article/pii/S1319157817300034` (cit. on pp. 1, 71).

[2] A. Gandomi and M. Haider, *Beyond the hype: Big data concepts, methods, and analytics*, Int J. Information Management **35** (2015) 137, URL: `https://doi.org/10.1016/j.ijinfomgt.2014.10.007` (cit. on pp. 1, 71).

[3] A. Doan, A. Y. Halevy and Z. G. Ives, *Principles of Data Integration*, Morgan Kaufmann, 2012, ISBN: 978-0-12-416044-6, URL: `http://research.cs.wisc.edu/dibook/` (cit. on pp. 1, 5, 13, 14, 17, 18, 26, 28, 42).

[4] P. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher and J. Darmont, *Metadata Systems for Data Lakes: Models and Features*, CoRR **abs/1909.09377** (2019), arXiv: `1909.09377`, URL: `http://arxiv.org/abs/1909.09377` (cit. on p. 2).

[5] F. Ravat and Y. Zhao, "Data Lakes: Trends and Perspectives", *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part I*, ed. by S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa and I. Khalil, vol. 11706, Lecture Notes in Computer Science, Springer, 2019 304, ISBN: 978-3-030-27614-0, URL: `https://doi.org/10.1007/978-3-030-27615-7%5C_23` (cit. on p. 2).

[6] S. Auer, S. Scerri, A. Versteden, E. Pauwels, A. Charalambidis, S. Konstantopoulos, J. Lehmann, H. Jabeen, I. Ermilov, G. Sejdiu, A. Ikonomopoulos, S. Andronopoulos, M. Vlachogiannis, C. Pappas, A. Davettas, I. A. Klampanos, E. Grigoropoulos, V. Karkaletsis, V. de Boer, R. Siebes, M. N. Mami, S. Albani, M. Lazzarini, P. Nunes, E. Angiuli, N. Pittaras, G. Giannakopoulos, G. Argyriou, G. Stamoulis, G. Papadakis, M. Koubarakis, P. Karampiperis, A. N. Ngomo and M. Vidal, "The BigDataEurope Platform - Supporting the Variety Dimension of Big Data", *Web Engineering - 17th International Conference, ICWE 2017, Rome, Italy, June 5-8, 2017, Proceedings*, ed. by J. Cabot, R. D. Virgilio and R. Torlone, vol. 10360, Lecture Notes in Computer Science, Springer, 2017 41, ISBN: 978-3-319-60130-4, URL: `https://doi.org/10.1007/978-3-319-60131-1%5C_3` (cit. on pp. 2, 17).

[7] M. K. Saggi and S. Jain, *A survey towards an integration of big data analytics to big insights for value-creation*, Inf. Process. Manage. **54** (2018) 758, URL: `https://doi.org/10.1016/j.ipm.2018.01.010` (cit. on p. 13).

[8] M. Lenzerini, "Data Integration: A Theoretical Perspective", *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, ed. by L. Popa, S. Abiteboul and P. G. Kolaitis, ACM, 2002 233, ISBN: 1-58113-507-6, URL: `https://doi.org/10.1145/543613.543644` (cit. on pp. 13, 14).

[9] M. Friedman, A. Y. Levy and T. D. Millstein, "Navigational Plans for Data Integration", *Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration, Held on July 31, 1999 in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence City Conference Center, Stockholm, Sweden*, vol. 23, CEUR Workshop Proceedings, CEUR-WS.org, 1999, URL: `http://ceur-ws.org/Vol-23/friedman-ijcai99-iii.ps` (cit. on p. 14).

[10] A. Y. Halevy, *Answering queries using views: A survey*, VLDB J. **10** (2001) 270, URL: `https://doi.org/10.1007/s007780100054` (cit. on p. 14).

[11] A. Y. Levy, A. Rajaraman and J. J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions", *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, ed. by T. M. Vijayaraman, A. P. Buchmann, C. Mohan and N. L. Sarda, Morgan Kaufmann, 1996 251, ISBN: 1-55860-382-4, URL: `http://www.vldb.org/conf/1996/P251.PDF` (cit. on p. 14).

[12] J. D. Ullman, "Information Integration Using Logical Views", *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, ed. by F. N. Afrati and P. G. Kolaitis, vol. 1186, Lecture Notes in Computer Science, Springer, 1997 19, ISBN: 3-540-62222-5, URL: `https://doi.org/10.1007/3-540-62222-5%5C_34` (cit. on p. 14).

[13] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems, Second Edition*, Prentice-Hall, 1999 (cit. on pp. 14, 15, 27).

[14] M. Chen, S. Mao and Y. Liu, *Big Data: A Survey*, MONET **19** (2014) 171, URL: `https://doi.org/10.1007/s11036-013-0489-0` (cit. on p. 16).

[15] I. G. Terrizzano, P. M. Schwarz, M. Roth and J. E. Colino, "Data Wrangling: The Challenging Yourney from the Wild to the Lake", *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*, www.cidrdb.org, 2015, URL: `http://cidrdb.org/cidr2015/Papers/CIDR15%5C_Paper2.pdf` (cit. on p. 17).

[16] T. Berners-Lee, *Linked Data*, `https://www.w3.org/DesignIssues/LinkedData.html`, [Online; accessed 19-July-2019], 2006 (cit. on p. 18).

[17] C. Bizer, T. Heath and T. Berners-Lee, *Linked Data - The Story So Far*, Int. J. Semantic Web Inf. Syst. **5** (2009) 1, URL: `https://doi.org/10.4018/jswis.2009081901` (cit. on p. 18).

[18] David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQueen, Henry S. Thompson, *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes.*, (2012), URL: `https://www.w3.org/TR/xmlschema11-2/.` (cit. on p. 18).

[19] M. D. A. Phillips, *Tags for Identifying Languages*, https://tools.ietf.org/html/rfc5646, [Online; accessed 19-July-2019], 2009 (cit. on p. 18).

[20] P. F. P.-S. Patrick J. Hayes, *RDF 1.1 Semantics.*, (2014), URL: http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/. (cit. on p. 18).

[21] J. Pérez, M. Arenas and C. Gutiérrez, *Semantics and complexity of SPARQL*, ACM Trans. Database Syst. **34** (2009) 16:1, URL: https://doi.org/10.1145/1567274.1567278 (cit. on pp. 18, 23).

[22] P. Hayes and P. Patel-Schneider, *RDF 1.1 semantics.*, (2014), URL: http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/. (cit. on p. 18).

[23] J. D. Fernández, A. Llaves and Ó. Corcho, "Efficient RDF Interchange (ERI) Format for RDF Data Streams", *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, ed. by P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. T. Groth, N. F. Noy, K. Janowicz and C. A. Goble, vol. 8797, Lecture Notes in Computer Science, Springer, 2014 244, ISBN: 978-3-319-11914-4, URL: https://doi.org/10.1007/978-3-319-11915-1%5C_16 (cit. on pp. 19, 42).

[24] A. S. E. Prud'hommeaux, *SPARQL Query Language for RDF.*, (2008), URL: http://www.w3.org/TR/rdf-sparql-query/. (cit. on p. 20).

[25] M. Arenas, C. Gutiérrez and J. Pérez, "Foundations of RDF Databases", *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, ed. by S. Tessaris, E. Franconi, T. Eiter, C. Gutiérrez, S. Handschuh, M. Rousset and R. A. Schmidt, vol. 5689, Lecture Notes in Computer Science, Springer, 2009 158, ISBN: 978-3-642-03753-5, URL: https://doi.org/10.1007/978-3-642-03754-2%5C_4 (cit. on pp. 20, 22).

[26] M. Schmidt, M. Meier and G. Lausen, "Foundations of SPARQL query optimization", *Database Theory - ICDT 2010, 13th International Conference, Lausanne, Switzerland, March 23-25, 2010, Proceedings*, ed. by L. Segoufin, ACM International Conference Proceeding Series, ACM, 2010 4, ISBN: 978-1-60558-947-3, URL: https://doi.org/10.1145/1804669.1804675 (cit. on pp. 21–23).

[27] M. Vidal, S. Castillo, M. Acosta, G. Montoya and G. Palma, *On the Selection of SPARQL Endpoints to Efficiently Execute Federated SPARQL Queries*, Trans. Large-Scale Data- and Knowledge-Centered Systems, Lecture Notes in Computer Science **25** (2016) 109, ed. by A. Hameurlain, J. Küng and R. R. Wagner, URL: https://doi.org/10.1007/978-3-662-49534-6%5C_4 (cit. on pp. 22, 33, 62).

[28] R. C. Souripriya Das Seema Sundara, *R2RML: RDB to RDF Mapping Language.*, (2012), URL: http://www.w3.org/TR/2012/REC-r2rml-20120927/. (cit. on p. 23).

[29] M. Saleem, Y. Khan, A. Hasnain, I. Ermilov and A. N. Ngomo, *A fine-grained evaluation of SPARQL endpoint federation systems*, Semantic Web **7** (2016) 493, URL: https://doi.org/10.3233/SW-150186 (cit. on pp. 27, 28).

[30] M. Acosta, O. Hartig and J. F. Sequeda, "Federated RDF Query Processing", *Encyclopedia of Big Data Technologies.* Ed. by S. Sakr and A. Y. Zomaya, Springer, 2019, ISBN: 978-3-319-63962-8, URL: `https://doi.org/10.1007/978-3-319-63962-8%5C_228-1` (cit. on p. 27).

[31] D. Oguz, B. Ergenc, S. Yin, O. Dikenelli and A. Hameurlain, *Federated query processing on linked data: a qualitative survey and open challenges*, Knowledge Eng. Review **30** (2015) 545, URL: `https://doi.org/10.1017/S0269888915000107` (cit. on pp. 28, 29).

[32] T. Ibaraki and T. Kameda, *On the Optimal Nesting Order for Computing N-Relational Joins*, ACM Trans. Database Syst. **9** (1984) 482, URL: `https://doi.org/10.1145/1270.1498` (cit. on p. 29).

[33] T. Urhan and M. J. Franklin, *XJoin: A Reactively-Scheduled Pipelined Join Operator*, IEEE Data Eng. Bull. **23** (2000) 27, URL: `http://sites.computer.org/debull/A00JUN-CD.pdf` (cit. on p. 29).

[34] M. Acosta, M. Vidal, T. Lampo, J. Castillo and E. Ruckhaus, "ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints", *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, ed. by L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy and E. Blomqvist, vol. 7031, Lecture Notes in Computer Science, Springer, 2011 18, ISBN: 978-3-642-25072-9, URL: `https://doi.org/10.1007/978-3-642-25073-6%5C_2` (cit. on pp. 29, 33, 36, 55–57, 62, 73, 78, 89).

[35] A. Schultz, A. Matteini, R. Isele, P. Mendes, C. Bizer and C. Becker, "LDIF-A Framework for Large-Scale Linked Data Integration", *21st International World Wide Web Conference (WWW2012)*, 2012 (cit. on p. 32).

[36] P. Haase, M. Schmidt and A. Schwarte, "The Information Workbench as a Self-Service Platform for Linked Data Applications", *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*, ed. by O. Hartig, A. Harth and J. F. Sequeda, vol. 782, CEUR Workshop Proceedings, CEUR-WS.org, 2011, URL: `http://ceur-ws.org/Vol-782/HaaseEtAl%5C_COLD2011.pdf` (cit. on p. 32).

[37] C. Morbidoni, D. L. Phuoc, A. Polleres, M. Samwald and G. Tummarello, "Previewing Semantic Web Pipes", *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, ed. by S. Bechhofer, M. Hauswirth, J. Hoffmann and M. Koubarakis, vol. 5021, Lecture Notes in Computer Science, Springer, 2008 843, ISBN: 978-3-540-68233-2, URL: `https://doi.org/10.1007/978-3-540-68234-9%5C_70` (cit. on p. 32).

[38] J. Michelfeit and T. Knap, "Linked Data Fusion in ODCleanStore", *Proceedings of the ISWC 2012 Posters & Demonstrations Track, Boston, USA, November 11-15, 2012*, ed. by B. Glimm and D. Huynh, vol. 914, CEUR Workshop Proceedings, CEUR-WS.org, 2012, URL: `http://ceur-ws.org/Vol-914/paper%5C_37.pdf` (cit. on p. 32).

[39] C. Bizer and A. Schultz,
"The R2R Framework: Publishing and Discovering Mappings on the Web", *Proceedings of the First International Workshop on Consuming Linked Data, Shanghai, China, November 8, 2010*, ed. by O. Hartig, A. Harth and J. F. Sequeda, vol. 665, CEUR Workshop Proceedings, CEUR-WS.org, 2010,
URL: `http://ceur-ws.org/Vol-665/BizerEtAl%5C_COLD2010.pdf`
(cit. on p. 32).

[40] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov,
"SILK - A Link Discovery Framework for the Web of Data", *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*
Ed. by C. Bizer, T. Heath, T. Berners-Lee and K. Idehen, vol. 538,
CEUR Workshop Proceedings, CEUR-WS.org, 2009,
URL: `http://ceur-ws.org/Vol-538/ldow2009%5C_paper13.pdf` (cit. on p. 32).

[41] P. N. Mendes, H. Mühleisen and C. Bizer, "Sieve: linked data quality assessment and fusion", *Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, March 30, 2012*, ed. by D. Srivastava and I. Ari, ACM, 2012 116, ISBN: 978-1-4503-1143-4,
URL: `https://doi.org/10.1145/2320765.2320803` (cit. on p. 32).

[42] S. Bischof, C. Martin, A. Polleres and P. Schneider,
"Collecting, Integrating, Enriching and Republishing Open City Data as Linked Data",
The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II,
ed. by M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, vol. 9367,
Lecture Notes in Computer Science, Springer, 2015 57, ISBN: 978-3-319-25009-0,
URL: `https://doi.org/10.1007/978-3-319-25010-6%5C_4` (cit. on p. 32).

[43] D. Florescu, A. Y. Levy and A. O. Mendelzon,
Database Techniques for the World-Wide Web: A Survey, SIGMOD Record **27** (1998) 59,
URL: `https://doi.org/10.1145/290593.290605` (cit. on p. 32).

[44] A. Y. Halevy, *Answering queries using views: A survey*, VLDB J. **10** (2001) 270,
URL: `https://doi.org/10.1007/s007780100054` (cit. on p. 32).

[45] A. Y. Halevy, A. Rajaraman and J. J. Ordille, "Data Integration: The Teenage Years", *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, ed. by U. Dayal, K. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha and Y. Kim, ACM, 2006 9, ISBN: 1-59593-385-9,
URL: `http://dl.acm.org/citation.cfm?id=1164130` (cit. on p. 32).

[46] Z. G. Ives, A. Y. Halevy, P. Mork and I. Tatarinov,
Piazza: mediation and integration infrastructure for Semantic Web data,
J. Web Semant. **1** (2004) 155,
URL: `https://doi.org/10.1016/j.websem.2003.11.003` (cit. on p. 32).

[47] V. Zadorozhny, L. Raschid, M. Vidal, T. Urhan and L. Bright,
"Efficient evaluation of queries in a mediator for WebSources",
Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002, ed. by M. J. Franklin, B. Moon and A. Ailamaki,

ACM, 2002 85, ISBN: 1-58113-497-5,
URL: `https://doi.org/10.1145/564691.564702` (cit. on pp. 32, 55, 94).

[48] C. Basca and A. Bernstein, *Querying a messy web of data with Avalanche*, J. Web Semant. **26** (2014) 1,
URL: `https://doi.org/10.1016/j.websem.2014.04.002` (cit. on p. 33).

[49] A. Charalambidis, A. Troumpoukis and S. Konstantopoulos, "SemaGrow: optimizing federated SPARQL queries", *Proceedings of the 11th International Conference on Semantic Systems, SEMANTICS 2015, Vienna, Austria, September 15-17, 2015*, ed. by A. Polleres, T. Pellegrini, S. Hellmann and J. X. Parreira, ACM, 2015 121,
ISBN: 978-1-4503-3462-4, URL: `https://doi.org/10.1145/2814864.2814886`
(cit. on pp. 33, 35, 36, 55).

[50] G. Montoya, H. Skaf-Molli and K. Hose, "The Odyssey Approach for Optimizing Federated SPARQL Queries", *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, ed. by C. d'Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange and J. Heflin, vol. 10587, Lecture Notes in Computer Science, Springer, 2017 471, ISBN: 978-3-319-68287-7,
URL: `https://doi.org/10.1007/978-3-319-68288-4%5C_28`
(cit. on pp. 33–36, 55).

[51] O. Görlitz and S. Staab, "SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions", *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*, ed. by O. Hartig, A. Harth and J. F. Sequeda, vol. 782, CEUR Workshop Proceedings, CEUR-WS.org, 2011,
URL: `http://ceur-ws.org/Vol-782/GoerlitzAndStaab%5C_COLD2011.pdf`
(cit. on pp. 33, 36, 55).

[52] A. Schwarte, P. Haase, K. Hose, R. Schenkel and M. Schmidt, "FedX: Optimization Techniques for Federated Query Processing on Linked Data", *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, ed. by L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy and E. Blomqvist, vol. 7031, Lecture Notes in Computer Science, Springer, 2011 601, ISBN: 978-3-642-25072-9,
URL: `https://doi.org/10.1007/978-3-642-25073-6%5C_38`
(cit. on pp. 33, 35, 36, 55, 57, 73, 81, 89).

[53] I. Abdelaziz, E. Mansour, M. Ouzzani, A. Aboulnaga and P. Kalnis, *Lusail: A System for Querying Linked Data at Scale*, PVLDB **11** (2017) 485,
URL: `http://www.vldb.org/pvldb/vol11/p485-abdelaziz.pdf`
(cit. on pp. 33, 35, 36, 55).

[54] R. Verborgh, M. V. Sande, O. Hartig, J. V. Herwegen, L. D. Vocht, B. D. Meester, G. Haesendonck and P. Colpaert, *Triple Pattern Fragments: A low-cost knowledge graph interface for the Web*, J. Web Semant. **37-38** (2016) 184,
URL: `https://doi.org/10.1016/j.websem.2016.03.003` (cit. on pp. 33, 35).

[55] M. Saleem and A. N. Ngomo,
"HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation",
The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings,
ed. by V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab and A. Tordai, vol. 8465, Lecture Notes in Computer Science, Springer, 2014 176, ISBN: 978-3-319-07442-9,
URL: `https://doi.org/10.1007/978-3-319-07443-6%5C_13` (cit. on pp. 33, 55).

[56] K. Alexander, R. Cyganiak, M. Hausenblas and J. Zhao, "Describing Linked Datasets", *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.* Ed. by C. Bizer, T. Heath, T. Berners-Lee and K. Idehen, vol. 538, CEUR Workshop Proceedings, CEUR-WS.org, 2009,
URL: `http://ceur-ws.org/Vol-538/ldow2009%5C_paper20.pdf` (cit. on p. 33).

[57] T. Neumann and G. Moerkotte,
"Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins", *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, ed. by S. Abiteboul, K. Böhm, C. Koch and K. Tan, IEEE Computer Society, 2011 984, ISBN: 978-1-4244-8958-9,
URL: `https://doi.org/10.1109/ICDE.2011.5767868` (cit. on p. 34).

[58] A. Gubichev and T. Neumann,
"Exploiting the query structure for efficient join ordering in SPARQL queries", *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014.* Ed. by S. Amer-Yahia, V. Christophides, A. Kementsietsidis, M. N. Garofalakis, S. Idreos and V. Leroy, OpenProceedings.org, 2014 439,
URL: `https://doi.org/10.5441/002/edbt.2014.40` (cit. on p. 34).

[59] M. Saleem, A. N. Ngomo, J. X. Parreira, H. F. Deus and M. Hauswirth,
"DAW: Duplicate-AWare Federated Query Processing over the Web of Data",
The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I, ed. by H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty and K. Janowicz, vol. 8218, Lecture Notes in Computer Science, Springer, 2013 574, ISBN: 978-3-642-41334-6,
URL: `https://doi.org/10.1007/978-3-642-41335-3%5C_36`
(cit. on pp. 34, 102).

[60] G. Montoya, H. Skaf-Molli, P. Molli and M. Vidal,
"Federated SPARQL Queries Processing with Replicated Fragments",
The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I,
ed. by M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, vol. 9366,
Lecture Notes in Computer Science, Springer, 2015 36, ISBN: 978-3-319-25006-9,
URL: `https://doi.org/10.1007/978-3-319-25007-6%5C_3` (cit. on pp. 34, 102).

[61] G. Montoya, M. Vidal and M. Acosta,
"A Heuristic-Based Approach for Planning Federated SPARQL Queries",
Proceedings of the Third International Workshop on Consuming Linked Data, COLD 2012, Boston, MA, USA, November 12, 2012, ed. by J. F. Sequeda, A. Harth and O. Hartig, vol. 905, CEUR Workshop Proceedings, CEUR-WS.org, 2012,

URL: http://ceur-ws.org/Vol-905/MontoyaEtAl%5C_COLD2012.pdf
(cit. on p. 35).

[62] B. Quilitz and U. Leser, "Querying Distributed RDF Data Sources with SPARQL", *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, ed. by S. Bechhofer, M. Hauswirth, J. Hoffmann and M. Koubarakis, vol. 5021, Lecture Notes in Computer Science, Springer, 2008 524, ISBN: 978-3-540-68233-2, URL: https://doi.org/10.1007/978-3-540-68234-9%5C_39 (cit. on p. 36).

[63] S. J. Lynden, I. Kojima, A. Matono and Y. Tanimura, "ADERIS: An Adaptive Query Processor for Joining Federated SPARQL Endpoints", *On the Move to Meaningful Internet Systems: OTM 2011 - Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part II*, ed. by R. Meersman, T. S. Dillon, P. Herrero, A. Kumar, M. Reichert, L. Qing, B. C. Ooi, E. Damiani, D. C. Schmidt, J. White, M. Hauswirth, P. Hitzler and M. K. Mohania, vol. 7045, Lecture Notes in Computer Science, Springer, 2011 808, ISBN: 978-3-642-25105-4, URL: https://doi.org/10.1007/978-3-642-25106-1%5C_28 (cit. on p. 36).

[64] C. Quix, R. Hai and I. Vatov, "GEMMS: A Generic and Extensible Metadata Management System for Data Lakes", *Proceedings of the CAiSE'16 Forum, at the 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016), Ljubljana, Slovenia, June 13-17, 2016.* Ed. by S. España, M. Ivanovic and M. Savic, vol. 1612, CEUR Workshop Proceedings, CEUR-WS.org, 2016 129, URL: http://ceur-ws.org/Vol-1612/paper17.pdf (cit. on p. 37).

[65] M. N. Mami, S. Scerri, S. Auer and M. Vidal, "Towards Semantification of Big Data Technology", *Big Data Analytics and Knowledge Discovery - 18th International Conference, DaWaK 2016, Porto, Portugal, September 6-8, 2016, Proceedings*, ed. by S. Madria and T. Hara, vol. 9829, Lecture Notes in Computer Science, Springer, 2016 376, ISBN: 978-3-319-43945-7, URL: https://doi.org/10.1007/978-3-319-43946-4%5C_25 (cit. on p. 37).

[66] C. Walker and H. H. Alrehamy, "Personal Data Lake with Data Gravity Pull", *Fifth IEEE International Conference on Big Data and Cloud Computing, BDCloud 2015, Dalian, China, August 26-28, 2015*, ed. by K. Li, H. Qi, J. Gaudiot, J. Kishigami, H. Wu, K. Li and Y. Wu, IEEE Computer Society, 2015 160, ISBN: 978-1-4673-7183-4, URL: https://doi.org/10.1109/BDCloud.2015.62 (cit. on p. 37).

[67] Y. Khan, A. Zimmermann, A. Jha, V. Gadepally, M. d'Aquin and R. Sahay, *One Size Does Not Fit All: Querying Web Polystores*, IEEE Access **7** (2019) 9598, URL: https://doi.org/10.1109/ACCESS.2018.2888601 (cit. on p. 37).

[68] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson and S. B. Zdonik, *The BigDAWG Polystore System*, SIGMOD Record **44** (2015) 11, URL: https://doi.org/10.1145/2814710.2814713 (cit. on p. 37).

[69] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati, *Authorization enforcement in distributed query evaluation*, Journal of Computer Security **19** (2011) 751, URL: `https://doi.org/10.3233/JCS-2010-0413` (cit. on p. 37).

[70] J. Bater, G. Elliott, C. Eggen, S. Goel, A. N. Kho and J. Rogers, *SMCQL: Secure Query Processing for Private Data Networks*, PVLDB **10** (2017) 673, URL: `http://www.vldb.org/pvldb/vol10/p673-rogers.pdf` (cit. on p. 37).

[71] L. Costabello, S. Villata and F. Gandon, "Context-Aware Access Control for RDF Graph Stores", *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, ed. by L. D. Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz and P. J. F. Lucas, vol. 242, Frontiers in Artificial Intelligence and Applications, IOS Press, 2012 282, ISBN: 978-1-61499-097-0, URL: `https://doi.org/10.3233/978-1-61499-098-7-282` (cit. on pp. 37, 87).

[72] Y. Khan, M. Saleem, M. Mehdi, A. Hogan, Q. Mehmood, D. Rebholz-Schuhmann and R. Sahay, *SAFE: SPARQL Federation over RDF Data Cubes with Access Control*, J. Biomedical Semantics **8** (2017) 5:1, URL: `https://doi.org/10.1186/s13326-017-0112-6` (cit. on pp. 37, 38, 87).

[73] J. Unbehauen, M. Frommhold and M. Martin, "Enforcing scalable authorization on SPARQL queries", *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016.* Ed. by M. Martin, M. Cuquet and E. Folmer, vol. 1695, CEUR Workshop Proceedings, CEUR-WS.org, 2016, URL: `http://ceur-ws.org/Vol-1695/paper38.pdf` (cit. on pp. 37, 38, 87).

[74] S. Kirrane, A. Abdelrahman, A. Mileo and S. Decker, "Secure Manipulation of Linked Data", *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, ed. by H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty and K. Janowicz, vol. 8218, Lecture Notes in Computer Science, Springer, 2013 248, ISBN: 978-3-642-41334-6, URL: `https://doi.org/10.1007/978-3-642-41335-3%5C_16` (cit. on pp. 37, 38).

[75] M. Amini and R. Jalili, *Multi-level authorisation model and framework for distributed semantic-aware environments*, IET Information Security **4** (2010) 301, URL: `https://doi.org/10.1049/iet-ifs.2009.0198` (cit. on pp. 37, 87).

[76] P. A. Bonatti and D. Olmedilla, "Rule-Based Policy Representation and Reasoning for the Semantic Web", *Reasoning Web, Third International Summer School 2007, Dresden, Germany, September 3-7, 2007, Tutorial Lectures*, ed. by G. Antoniou, U. Aßmann, C. Baroglio, S. Decker, N. Henze, P. Patranjan and R. Tolksdorf, vol. 4636, Lecture Notes in Computer Science, Springer, 2007 240, ISBN: 978-3-540-74613-3, URL: `https://doi.org/10.1007/978-3-540-74615-7%5C_4` (cit. on p. 37).

[77] H. Mühleisen, M. Kost and J.-C. Freytag, *SWRL-based Access Policies for Linked Data*, Proceedings of the 2nd Workshop on Trust and Privacy on the Social and Semantic Web (SPOT-2010) **80** (2010) (cit. on p. 37).

[78] L. Pellegrino, F. Huet, F. Baude and A. Alshabani, "A Distributed Publish/Subscribe System for RDF Data", *Data Management in Cloud, Grid and P2P Systems - 6th International Conference, Globe 2013, Prague, Czech Republic, August 28-29, 2013. Proceedings*, ed. by A. Hameurlain, J. W. Rahayu and D. Taniar, vol. 8059, Lecture Notes in Computer Science, Springer, 2013 39, ISBN: 978-3-642-40052-0, URL: `https://doi.org/10.1007/978-3-642-40053-7%5C_4` (cit. on p. 38).

[79] P. Chirita, S. Idreos, M. Koubarakis and W. Nejdl, "Publish/Subscribe for RDF-based P2P Networks", *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, ed. by C. Bussler, J. Davies, D. Fensel and R. Studer, vol. 3053, Lecture Notes in Computer Science, Springer, 2004 182, ISBN: 3-540-21999-4, URL: `https://doi.org/10.1007/978-3-540-25956-5%5C_13` (cit. on p. 38).

[80] N. Popitsch and B. Haslhofer, *DSNotify - A solution for event detection and link maintenance in dynamic datasets*, J. Web Semant. **9** (2011) 266, URL: `https://doi.org/10.1016/j.websem.2011.05.002` (cit. on p. 38).

[81] S. Tramp, P. Frischmuth, T. Ermilov and S. Auer, "Weaving a Social Data Web with Semantic Pingback", *Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings*, ed. by P. Cimiano and H. S. Pinto, vol. 6317, Lecture Notes in Computer Science, Springer, 2010 135, ISBN: 978-3-642-16437-8, URL: `https://doi.org/10.1007/978-3-642-16438-5%5C_10` (cit. on p. 38).

[82] G. Tummarello, C. Morbidoni, R. Bachmann-Gmür and O. Erling, "RDFSync: Efficient Remote Synchronization of RDF Models", *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.* Ed. by K. Aberer, K. Choi, N. F. Noy, D. Allemang, K. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber and P. Cudré-Mauroux, vol. 4825, Lecture Notes in Computer Science, Springer, 2007 537, ISBN: 978-3-540-76297-3, URL: `https://doi.org/10.1007/978-3-540-76298-0%5C_39` (cit. on p. 38).

[83] B. Schandl, "Replication and Versioning of Partial RDF Graphs", *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, ed. by L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral and T. Tudorache, vol. 6088, Lecture Notes in Computer Science, Springer, 2010 31, ISBN: 978-3-642-13485-2, URL: `https://doi.org/10.1007/978-3-642-13486-9%5C_3` (cit. on p. 38).

[84] K. Voruganti, M. T. Özsu and R. C. Unrau, *An Adaptive Data-Shipping Architecture for Client Caching Data Management Systems*, Distributed and Parallel Databases **15** (2004) 137, URL: `https://doi.org/10.1023/B:DAPD.0000013069.97679.62` (cit. on p. 38).

[85] Y. Breitbart, R. Komondoor, R. Rastogi, S. Seshadri and A. Silberschatz, "Update Propagation Protocols For Replicated Databases", *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.* Ed. by A. Delis, C. Faloutsos and S. Ghandeharizadeh, ACM Press, 1999 97, ISBN: 1-58113-084-8, URL: `https://doi.org/10.1145/304182.304191` (cit. on p. 38).

[86] A. Passant and P. N. Mendes, "sparqlPuSH: Proactive Notification of Data Updates in RDF Stores Using PubSubHubbub", *Proceedings of the Sixth Workshop on Scripting and Development for the Semantic Web, Crete, Greece, May 31, 2010*, ed. by G. A. Grimnes, S. Auer and G. T. Williams, vol. 699, CEUR Workshop Proceedings, CEUR-WS.org, 2010, URL: `http://ceur-ws.org/Vol-699/Paper6.pdf` (cit. on p. 38).

[87] J. Bleiholder, *Data fusion and conflict resolution in integrated information systems*, PhD thesis: University of Potsdam, 2010, URL: `http://d-nb.info/101910807X` (cit. on pp. 38, 115).

[88] K. M. Endris, M. Vidal and S. Auer, "FedSDM: Semantic Data Manager for Federations of RDF Datasets", *Data Integration in the Life Sciences - 13th International Conference, DILS 2018, Hannover, Germany, November 20-21, 2018, Proceedings*, ed. by S. Auer and M. Vidal, vol. 11371, Lecture Notes in Computer Science, Springer, 2018 85, ISBN: 978-3-030-06015-2, URL: `https://doi.org/10.1007/978-3-030-06016-9%5C_8` (cit. on p. 39).

[89] K. M. Endris, M. Galkin, I. Lytra, M. N. Mami, M. Vidal and S. Auer, "MULDER: Querying the Linked Data Web by Bridging RDF Molecule Templates", *Database and Expert Systems Applications - 28th International Conference, DEXA 2017, Lyon, France, August 28-31, 2017, Proceedings, Part I*, 2017 3, URL: `https://doi.org/10.1007/978-3-319-64468-4%5C_1` (cit. on pp. 39, 55, 89).

[90] K. M. Endris, M. Galkin, I. Lytra, M. N. Mami, M. Vidal and S. Auer, *Querying Interlinked Data by Bridging RDF Molecule Templates*, T. Large-Scale Data- and Knowledge-Centered Systems, Lecture Notes in Computer Science **39** (2018) 1, ed. by A. Hameurlain, R. R. Wagner, D. Benslimane, E. Damiani and W. I. Grosky, URL: `https://doi.org/10.1007/978-3-662-58415-6%5C_1` (cit. on pp. 39, 55, 73, 81).

[91] K. M. Endris, P. D. Rohde, M. Vidal and S. Auer, "Ontario: Federated Query Processing Against a Semantic Data Lake", *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part I*, ed. by S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa and I. Khalil, vol. 11706, Lecture Notes in Computer Science, Springer, 2019 379, ISBN: 978-3-030-27614-0, URL: `https://doi.org/10.1007/978-3-030-27615-7%5C_29` (cit. on pp. 39, 71).

[92] C. Bizer and A. Schultz, *The Berlin SPARQL Benchmark*, Int. J. Semantic Web Inf. Syst. **5** (2009) 1, URL: `https://doi.org/10.4018/jswis.2009040101` (cit. on pp. 47, 62).

[93] A. Hasnain, Q. Mehmood, S. S. e Zainab, M. Saleem, C. N. W. Jr., D. Zehra, S. Decker and D. Rebholz-Schuhmann, *BioFed: federated query processing over life sciences linked open data*, J. Biomedical Semantics **8** (2017) 13:1, URL: `https://doi.org/10.1186/s13326-017-0118-0` (cit. on pp. 49, 63, 81).

[94] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte and T. Tran, "FedBench: A Benchmark Suite for Federated Semantic Data Query Processing", *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, ed. by L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy and E. Blomqvist, vol. 7031, Lecture Notes in Computer Science, Springer, 2011 585, ISBN: 978-3-642-25072-9, URL: `https://doi.org/10.1007/978-3-642-25073-6%5C_37` (cit. on pp. 50, 56, 62).

[95] M. Schmidt, T. Hornung, M. Meier, C. Pinkel and G. Lausen, "SP^{2}Bench: A SPARQL Performance Benchmark", *Semantic Web Information Management - A Model-Based Perspective*, ed. by R. D. Virgilio, F. Giunchiglia and L. Tanca, Springer, 2009 371, ISBN: 978-3-642-04328-4, URL: `https://doi.org/10.1007/978-3-642-04329-1%5C_16` (cit. on p. 50).

[96] C. Chen, B. Golshan, A. Y. Halevy, W. Tan and A. Doan, *BigGorilla: An Open-Source Ecosystem for Data Preparation and Integration*, IEEE Data Eng. Bull. **41** (2018) 10, URL: `http://sites.computer.org/debull/A18june/p10.pdf` (cit. on p. 55).

[97] A. Doan and A. Y. Halevy, *Semantic Integration Research in the Database Community: A Brief Survey*, AI Magazine **26** (2005) 83, URL: `http://www.aaai.org/ojs/index.php/aimagazine/article/view/1801` (cit. on p. 55).

[98] Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy and D. S. Weld, "An Adaptive Query Execution System for Data Integration", *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.* Ed. by A. Delis, C. Faloutsos and S. Ghandeharizadeh, ACM Press, 1999 299, ISBN: 1-58113-084-8, URL: `https://doi.org/10.1145/304182.304209` (cit. on p. 55).

[99] W. Scheufele and G. Moerkotte, "On the Complexity of Generating Optimal Plans with Cross Products", *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA*, ed. by A. O. Mendelzon and Z. M. Özsoyoglu, ACM Press, 1997 238, ISBN: 0-89791-910-6, URL: `https://doi.org/10.1145/263661.263687` (cit. on p. 61).

[100] M. Acosta, M. Vidal and Y. Sure-Vetter, "Diefficiency Metrics: Measuring the Continuous Efficiency of Query Processing Approaches", *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, ed. by C. d'Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange and J. Heflin, vol. 10588,

Lecture Notes in Computer Science, Springer, 2017 3, ISBN: 978-3-319-68203-7,
URL: `https://doi.org/10.1007/978-3-319-68204-4%5C_1` (cit. on p. 66).

[101] C. Weiss, P. Karras and A. Bernstein,
Hexastore: sextuple indexing for semantic web data management, PVLDB **1** (2008) 1008,
URL: `http://www.vldb.org/pvldb/1/1453965.pdf` (cit. on p. 77).

[102] K. M. Endris, Z. Almhithawi, I. Lytra, M. Vidal and S. Auer,
"BOUNCER: Privacy-Aware Query Processing over Federations of RDF Datasets",
Database and Expert Systems Applications - 29th International Conference, DEXA 2018, Regensburg, Germany, September 3-6, 2018, Proceedings, Part I, 2018 69,
URL: `https://doi.org/10.1007/978-3-319-98809-2%5C_5` (cit. on p. 87).

[103] M. Vidal, E. Ruckhaus, T. Lampo, A. Martínez, J. Sierra and A. Polleres,
"Efficiently Joining Group Patterns in SPARQL Queries",
The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I, ed. by L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral and T. Tudorache, vol. 6088, Lecture Notes in Computer Science, Springer, 2010 228, ISBN: 978-3-642-13485-2,
URL: `https://doi.org/10.1007/978-3-642-13486-9%5C_16` (cit. on p. 91).

[104] R. Verborgh, O. Hartig, B. D. Meester, G. Haesendonck, L. D. Vocht, M. V. Sande, R. Cyganiak, P. Colpaert, E. Mannens and R. V. de Walle,
"Querying Datasets on the Web with High Availability",
The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I, ed. by P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. T. Groth, N. F. Noy, K. Janowicz and C. A. Goble, vol. 8796, Lecture Notes in Computer Science, Springer, 2014 180, ISBN: 978-3-319-11963-2,
URL: `https://doi.org/10.1007/978-3-319-11964-9%5C_12`
(cit. on pp. 101, 102).

[105] K. M. Endris, S. Faisal, F. Orlandi, S. Auer and S. Scerri,
"Interest-Based RDF Update Propagation", *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, ed. by M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, vol. 9366,
Lecture Notes in Computer Science, Springer, 2015 513, ISBN: 978-3-319-25006-9,
URL: `https://doi.org/10.1007/978-3-319-25007-6%5C_30` (cit. on p. 101).

[106] K. M. Endris, S. Faisal, F. Orlandi, S. Auer and S. Scerri,
"iRap - an Interest-Based RDF Update Propagation Framework",
Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015. Ed. by S. Villata, J. Z. Pan and M. Dragoni, vol. 1486, CEUR Workshop Proceedings, CEUR-WS.org, 2015, URL: `http://ceur-ws.org/Vol-1486/paper%5C_124.pdf`
(cit. on pp. 101, 120).

[107] S. Faisal, K. M. Endris, S. Shekarpour, S. Auer and M. Vidal, "Co-evolution of RDF Datasets",
Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, ed. by A. Bozzon, P. Cudré-Mauroux and C. Pautasso, vol. 9671,
Lecture Notes in Computer Science, Springer, 2016 225, ISBN: 978-3-319-38790-1,
URL: `https://doi.org/10.1007/978-3-319-38791-8%5C_13` (cit. on p. 101).

[108] L. D. Ibáñez, H. Skaf-Molli, P. Molli and O. Corby,
"Col-Graph: Towards Writable and Scalable Linked Open Data",
The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I, ed. by P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. T. Groth, N. F. Noy, K. Janowicz and C. A. Goble, vol. 8796, Lecture Notes in Computer Science, Springer, 2014 325, ISBN: 978-3-319-11963-2, URL: `https://doi.org/10.1007/978-3-319-11964-9%5C_21` (cit. on p. 102).

[109] E. Marx, S. Shekarpour, S. Auer and A. N. Ngomo, "Large-Scale RDF Dataset Slicing", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, IEEE Computer Society, 2013 228, ISBN: 978-0-7695-5119-7, URL: `https://doi.org/10.1109/ICSC.2013.47` (cit. on p. 103).

[110] *SPARQL 1.1 Query Language*,
`http://www.w3.org/TR/2013/REC-sparql11-query-20130321/`, 2013 (cit. on p. 105).

[111] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann and S. Auer,
Quality assessment for Linked Data: A Survey, Semantic Web **7** (2016) 63,
URL: `https://doi.org/10.3233/SW-150175` (cit. on p. 121).

[112] H. Zafar, G. Napolitano and J. Lehmann,
"Formal Query Generation for Question Answering over Knowledge Bases",
The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings, ed. by A. Gangemi, R. Navigli, M. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, vol. 10843, Lecture Notes in Computer Science, Springer, 2018 714, ISBN: 978-3-319-93416-7,
URL: `https://doi.org/10.1007/978-3-319-93417-4%5C_46` (cit. on p. 126).

[113] S. Bechhofer, M. Hauswirth, J. Hoffmann and M. Koubarakis, eds.,
The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings, vol. 5021, Lecture Notes in Computer Science, Springer, 2008, ISBN: 978-3-540-68233-2,
URL: `https://doi.org/10.1007/978-3-540-68234-9`.

[114] P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. T. Groth, N. F. Noy, K. Janowicz and C. A. Goble, eds.,
The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I, vol. 8796, Lecture Notes in Computer Science, Springer, 2014, ISBN: 978-3-319-11963-2,
URL: `https://doi.org/10.1007/978-3-319-11964-9`.

[115] L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral and T. Tudorache, eds., *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, vol. 6088, Lecture Notes in Computer Science, Springer, 2010, ISBN: 978-3-642-13485-2,
URL: `https://doi.org/10.1007/978-3-642-13486-9`.

[116] S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa and I. Khalil, eds.,
Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part I, vol. 11706,

Lecture Notes in Computer Science, Springer, 2019, ISBN: 978-3-030-27614-0, URL: https://doi.org/10.1007/978-3-030-27615-7.

[117] A. Delis, C. Faloutsos and S. Ghandeharizadeh, eds., *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, ACM Press, 1999, ISBN: 1-58113-084-8.

[118] L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy and E. Blomqvist, eds., *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, vol. 7031, Lecture Notes in Computer Science, Springer, 2011, ISBN: 978-3-642-25072-9, URL: https://doi.org/10.1007/978-3-642-25073-6.

[119] M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, eds., *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, vol. 9366, Lecture Notes in Computer Science, Springer, 2015, ISBN: 978-3-319-25006-9, URL: https://doi.org/10.1007/978-3-319-25007-6.

[120] C. Bizer, T. Heath, T. Berners-Lee and K. Idehen, eds., *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*, vol. 538, CEUR Workshop Proceedings, CEUR-WS.org, 2009, URL: http://ceur-ws.org/Vol-538.

[121] H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty and K. Janowicz, eds., *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, vol. 8218, Lecture Notes in Computer Science, Springer, 2013, ISBN: 978-3-642-41334-6, URL: https://doi.org/10.1007/978-3-642-41335-3.

[122] O. Hartig, A. Harth and J. F. Sequeda, eds., *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*, vol. 782, CEUR Workshop Proceedings, CEUR-WS.org, 2011, URL: http://ceur-ws.org/Vol-782.

APPENDIX **A**

List of Publications

A.1 Publications

The following publications have been produced during the work on this thesis. These articles have been presented and published in the proceedings of the conferences and journals.

- *Conference Papers*
  1. **Kemele M Endris**, Philipp D. Rohde, Maria-Esther Vidal, Sören Auer, *Ontario: Federated Query Processing against Heterogeneous Data Sources in a Data Lake.* International Conference on Database and Expert Systems Applications, DEXA 2019.
  2. **Kemele M. Endris**, Mikhail Galkin, Ioanna Lytra, Mohamed Nadjib Mami, Maria-Esther Vidal, Sören Auer, *MULDER: Querying the Linked Data Web by Bridging RDF Molecule Templates.* In International Conference on Database and Expert Systems Applications, pp. 3-18. Springer, Cham, 2017. **Best Paper Award**
  3. **Kemele M. Endris**, Mikhail Galkin, Ioanna Lytra, Mohamed Nadjib Mami, Maria-Esther Vidal, Sören Auer, *Querying Interlinked Data by Bridging RDF Molecule Templates.* In Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIX, pp. 1-42. Springer, Berlin, Heidelberg, 2018.
  4. **Kemele M Endris**, Zuhair Almhithawi, Ioanna Lytra, Maria-Esther Vidal, Sören Auer, *BOUNCER: Privacy-Aware Query Processing over Federations of RDF Datasets.* In International Conference on Database and Expert Systems Applications, pp. 69-84. Springer, Cham, 2018.
  5. **Kemele M. Endris**, Sidra Faisal, Fabrizio Orlandi, Sören Auer, Simon Scerri, *Interest-based RDF update propagation*, In Proceedings of the 14th International Conference on The Semantic Web-ISWC 2015-Volume 9366, pp. 513-529. Springer-Verlag, 2015.
  6. **Kemele M. Endris**, José M. Giménez-García, Harsh Thakkar, Elena Demidova, Antoine Zimmermann, Christoph Lange, Elena Simperl: *Dataset Reuse: An Analysis of References in Community Discussions, Publications and Data.* Short paper in International Conference on Knowledge Capture (K-CAP) 2017: 5:1-5:4
  7. Lucie-Aimée Kaffee, **Kemele M. Endris**, Elena Simperl and Maria-Esther Vidal, *Ranking Knowledge Graphs By Capturing Knowledge about Languages and Labels*, Proceedings of the Knowledge Capture Conference (K-CAP) 2019, Marina del Rey, California, USA, ACM 2019

8. Lucie-Aimée Kaffee, **Kemele M. Endris**, Elena Simperl "When humans and machines collaborate: cross-lingual label editing in wikidata." Proceedings of the 15th International Symposium on Open Collaboration. ACM, 2019.
9. David Chaves-Fraga, **Kemele Endris**, Enrique Iglesias, Oscar Corcho and Maria-Esther Vidal, *What are the Parameters that Affect the Construction of a Knowledge Graph?*, Proceedings of the 18th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE) (in OTM) 2019, Rhodes, Greece, Springer 2019.
10. Sidra Faisal, **Kemele M. Endris**, Saeedeh Shekarpour, Sören Auer, Maria-Esther Vidal, *Co-evolution of RDF Datasets*, Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, pp. 225–243, 2016
11. Mikhail Galkin, **Kemele M. Endris**, Maribel Acosta, Diego Collarana, Maria-Esther Vidal, Sören Auer. *SMJoin: A Multi-way Join Operator for SPARQL Queries.* In Proceedings of the 13th International Conference on Semantic Systems (SEMANTiCS) 2017, 104–111, Springer;
12. Saeedeh Shekarpour, **Kemele M. Endris**, Ashwini Jaya Kumar, Denis Lukovnikov, Kuldeep Singh, Harsh Thakkar, Christoph Lange: *Question Answering on Linked Data: Challenges and Future Directions.* WWW (Companion Volume) 2016: 693-698
13. Harsh Thakkar, **Kemele M. Endris**, José M. Giménez-García, Jeremy Debattista, Christoph Lange, Sören Auer: *Are Linked Datasets fit for Open-domain Question Answering? A Quality Assessment.* WIMS 2016: 19:1-19:12
14. Omar Al-Safi, Christian Mader, Ioanna Lytra, Mikhail Galkin, **Kemele M. Endris**, Maria-Esther Vidal, Sören Auer: *Shipping Knowledge Graph Management Capabilities to Data Providers and Consumers.* ICSC 2018: 9-16
15. Marlene Goncalves, Maria-Esther Vidal, **Kemele M. Endris**, *PURE: A Privacy Aware Rule-Based Framework over Knowledge Graphs.* Special paper in International Conference on Database and Expert Systems Applications, DEXA 2019.

- *Journal Articles*

16. **Kemele M. Endris**, Mikhail Galkin, Ioanna Lytra, Mohamed Nadjib Mami, Maria-Esther Vidal, Sören Auer. *MULDER: Querying the Linked Data Web by Bridging RDF Molecule Templates.* Accepted for publication in the LNCS Transactions on Large-Scale Data- and Knowledge-Centered Systems Journal (Transactions LDKS), 2018;
17. Maria-Esther Vidal, **Kemele M. Endris**, Samaneh Jazashoori, Ahmad Sakor, Ariam Rivas: *Transforming Heterogeneous Data into Knowledge for Personalized Treatments - A Use Case.* Datenbank-Spektrum 19(2): 95-106 (2019)

- *Book Chapters*

18. Maria-Esther Vidal, **Kemele M Endris**, Samaneh Jozashoori, Farah Karim, Guillermo Palma: *Semantic data integration of big biomedical data for supporting personalised medicine*, Current Trends in Semantic Web Technologies: Theory and Practice, Springer, Cham, 25-56 (2019)

- *Demos and Posters*

19. **Kemele M. Endris**, Sidra Faisal, Fabrizio Orlandi, Sören Auer, Simon Scerri, *iRap-an Interest-Based RDF Update Propagation Framework*. Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.
20. **Kemele M Endris**, Maria-Esther Vidal, Sören Auer, *FedSDM: Semantic Data Manager for Federations of RDF Datasets.* In International Conference on Data Integration in the Life Sciences, pp. 85-90. Springer, Cham, 2018.
21. Maria-Esther Vidal, **Kemele M. Endris**, Samaneh Jozashoori, Guillermo Palma: *A Knowledge Driven Pipeline for Transforming Big Data into Actionable Knowledge.* In International Conference on Data Integration in the Life Sciences (DILS) 2018: 44-49

APPENDIX B

Benchmark Queries

B.1 BSBM Queries

Listing B.1: Prefixes

```
PREFIX bsbm-inst: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rev:  <http://purl.org/stuff/rev#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
```

Listing B.2: **B1**: Find products for a given set of generic features

```
SELECT DISTINCT ?product ?label
WHERE {
     ?product rdfs:label ?label .
     ?product a %ProductType% .
     ?product bsbm:productFeature %ProductFeature1% .
     ?product bsbm:productFeature %ProductFeature2% .
     ?product bsbm:productPropertyNumeric1 ?value1 .
     FILTER (?value1 > %x%)
}
```

Listing B.3: **B2**: Retrieve basic information about a specific product for display purposes

```
SELECT ?label ?comment ?producer ?productFeature ?propertyTextual1 ?
    propertyTextual2 ?propertyTextual3
 ?propertyNumeric1 ?propertyNumeric2 ?propertyTextual4 ?propertyTextual5 ?
     propertyNumeric4
WHERE {
    %ProductXYZ% rdfs:label ?label .
    %ProductXYZ% rdfs:comment ?comment .
    %ProductXYZ% bsbm:producer ?p .
    ?p rdfs:label ?producer .
    %ProductXYZ% dc:publisher ?p .
    %ProductXYZ% bsbm:productFeature ?f .
    ?f rdfs:label ?productFeature .
```

```
        %ProductXYZ% bsbm:productPropertyTextual1 ?propertyTextual1 .
        %ProductXYZ% bsbm:productPropertyTextual2 ?propertyTextual2 .
        %ProductXYZ% bsbm:productPropertyTextual3 ?propertyTextual3 .
        %ProductXYZ% bsbm:productPropertyNumeric1 ?propertyNumeric1 .
        %ProductXYZ% bsbm:productPropertyNumeric2 ?propertyNumeric2 .
        OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual4 ?propertyTextual4 }
        OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual5 ?propertyTextual5 }
        OPTIONAL { %ProductXYZ% bsbm:productPropertyNumeric4 ?propertyNumeric4 }
    }
```

Listing B.4: **B3**: Find products having some specific features and not having one feature

```
    SELECT ?product ?label
    WHERE {
        ?product rdfs:label ?label .
        ?product a %ProductType% .
        ?product bsbm:productFeature %ProductFeature1% .
        ?product bsbm:productPropertyNumeric1 ?p1 .
        ?product bsbm:productPropertyNumeric3 ?p3 .
        ?product bsbm:productFeature ?pf .
        FILTER (?p3 < %y% )
        FILTER ( ?p1 > %x% )
        FILTER (?pf != bsbm-inst:ProductFeature8001)
    }
```

Listing B.5: **B4**: Find products matching two different sets of features

```
    SELECT DISTINCT ?product ?label ?propertyTextual
    WHERE {
     {
        ?product rdfs:label ?label .
        ?product rdf:type %ProductType% .
        ?product bsbm:productFeature %ProductFeature1% .
        ?product bsbm:productFeature %ProductFeature2% .
        ?product bsbm:productPropertyTextual1 ?propertyTextual .
        ?product bsbm:productPropertyNumeric1 ?p1 .
        FILTER ( ?p1 > %x% )
     } UNION {
        ?product rdfs:label ?label .
        ?product rdf:type %ProductType% .
        ?product bsbm:productFeature %ProductFeature1% .
        ?product bsbm:productFeature %ProductFeature3% .
        ?product bsbm:productPropertyTextual1 ?propertyTextual .
        ?product bsbm:productPropertyNumeric2 ?p2 .
        FILTER ( ?p2> %y% )
     }
    }
```

Listing B.6: **B5**: Find product that are similar to a given product

```
    SELECT DISTINCT ?product ?productLabel
    WHERE {
      ?product rdfs:label ?productLabel .
      %ProductXYZ% bsbm:productFeature ?prodFeature .
```

```
  ?product bsbm:productFeature ?prodFeature .
  %ProductXYZ% bsbm:productPropertyNumeric1 ?origProperty1 .
  ?product bsbm:productPropertyNumeric1 ?simProperty1 .
  %ProductXYZ% bsbm:productPropertyNumeric2 ?origProperty2 .
  ?product bsbm:productPropertyNumeric2 ?simProperty2 .
  FILTER (%ProductXYZ% != ?product)
  FILTER (?simProperty1 < (?origProperty1 + 120) && ?simProperty1 > (?
      origProperty1 - 120))
  FILTER (?simProperty2 < (?origProperty2 + 170) && ?simProperty2 > (?
      origProperty2 - 170))
}
```

Listing B.7: **B6**: Retrieve in-depth information about a specific product including offers and reviews

```
SELECT ?productLabel ?offer ?price ?vendor ?vendorTitle ?review ?revTitle ?
    reviewer
        ?revName ?rating1 ?rating2
WHERE {
    %ProductXYZ% rdfs:label ?productLabel .
    OPTIONAL {
        ?offer bsbm:product %ProductXYZ% .
      ?offer bsbm:price ?price .
      ?offer bsbm:vendor ?vendor .
      ?vendor rdfs:label ?vendorTitle .
        ?vendor bsbm:country <http://downlode.org/rdf/iso-3166/countries#DE> .
        ?offer dc:publisher ?vendor .
        ?offer bsbm:validTo ?date .
        FILTER (?date > %currentDate% )
    }
    OPTIONAL {
      ?review bsbm:reviewFor %ProductXYZ% .
      ?review rev:reviewer ?reviewer .
      ?reviewer foaf:name ?revName .
      ?review dc:title ?revTitle .
        OPTIONAL { ?review bsbm:rating1 ?rating1 . }
        OPTIONAL { ?review bsbm:rating2 ?rating2 . }
    }
}
```

Listing B.8: **B7**: Give me recent reviews in English for a specific product

```
SELECT ?title ?text ?reviewDate ?reviewer ?reviewerName ?rating1 ?rating2 ?
    rating3 ?rating4
WHERE {
  ?review bsbm:reviewFor %ProductXYZ% .
  ?review dc:title ?title .
  ?review rev:text ?text .
  FILTER langMatches( lang(?text), "EN" )
  ?review bsbm:reviewDate ?reviewDate .
  ?review rev:reviewer ?reviewer .
  ?reviewer foaf:name ?reviewerName .
  OPTIONAL { ?review bsbm:rating1 ?rating1 . }
  OPTIONAL { ?review bsbm:rating2 ?rating2 . }
  OPTIONAL { ?review bsbm:rating3 ?rating3 . }
  OPTIONAL { ?review bsbm:rating4 ?rating4 . }
}
```

Listing B.9: **B8**: Get offers for a given product which fulfill specific requirements

```
SELECT DISTINCT ?offer ?price
WHERE {
    ?offer bsbm:product %ProductXYZ% .
    ?offer bsbm:vendor ?vendor .
    ?offer dc:publisher ?vendor .
    ?vendor bsbm:country <http://downlode.org/rdf/iso-3166/countries#US> .
    ?offer bsbm:deliveryDays ?deliveryDays .
    ?offer bsbm:price ?price .
    ?offer bsbm:validTo ?date .
    FILTER (?date > %currentDate% )
    FILTER (?deliveryDays <= 3)
}
```

Listing B.10: **B9**: Get all information about an offer.

```
SELECT ?property ?hasValue ?isValueOf
WHERE {
 { %OfferXYZ% ?property ?hasValue }
 UNION { ?isValueOf ?property %OfferXYZ% }
}
```

Listing B.11: **B10**: Get all products' review text and product label from a specific producer

```
SELECT DISTINCT ?product ?revText ?rating3 ?plabel
WHERE {
    ?product rdfs:label ?plabel .
    ?product bsbm:producer %producer1% .
    ?product a ?productType .
    %producer1% a bsbm:Producer .
    %producer1%  rdfs:label ?prlabel .
    %producer1%  foaf:homepage ?homepage .
    ?review bsbm:reviewFor  ?product .
    ?review bsbm:rating1 ?rating1 .
    ?review bsbm:rating2 ?rating2 .
    ?review bsbm:rating3 ?rating3 .
    ?review rev:text ?revText .
}
```

Listing B.12: **B11**: Find offer and vendors of a product with a given feature and type

```
SELECT DISTINCT ?product ?producer ?offer ?vendor
WHERE {
    ?product a %ProductType1% .
    ?product rdfs:label ?label .
    ?product bsbm:productFeature %ProductFeature1% .
    ?product rdfs:comment ?productComment .
    ?product bsbm:producer ?producer .
    ?product dc:publisher ?publisher .
    ?product bsbm:productPropertyTextual1 ?propertyTextual1 .
    ?product bsbm:productPropertyTextual2 ?propertyTextual2 .
    ?product bsbm:productPropertyTextual3 ?propertyTextual3 .
    ?product bsbm:productPropertyNumeric1 ?propertyNumeric1 .
    ?product bsbm:productPropertyNumeric2 ?propertyNumeric2 .
    ?producer rdfs:label ?producerLabel .
    ?producer rdfs:comment ?producerComment.
    ?producer dc:publisher ?producerPublisher.
    ?offer bsbm:product ?product .
    ?offer bsbm:price ?price .
    ?offer bsbm:vendor ?vendor .
    ?offer bsbm:validTo %currentDate% .
    ?offer bsbm:validFrom ?offerValidFrom.
    ?offer bsbm:deliveryDays ?offerDeliveryDays.
    ?offer dc:publisher ?offerPublisher.
    ?offer dc:date ?offerPublishDate.
    ?vendor rdfs:label ?vendorLabel.
    ?vendor rdfs:comment ?vendorComment.
    ?vendor bsbm:country ?vcountry.
}
```

Listing B.13: **B12**: Find information about its feature, review and reviewer about a given product

```
SELECT DISTINCT ?label ?productType ?productFeature ?producer ?review ?reviewer
WHERE {
    %Product1% a  ?productType .
    %Product1% rdfs:label ?label .
    %Product1% bsbm:productFeature ?productFeature .
    %Product1% rdfs:comment ?productComment .
    %Product1% bsbm:producer ?producer .
    %Product1% dc:publisher ?publisher .
    %Product1% bsbm:productPropertyTextual1 ?propertyTextual1 .
    %Product1% bsbm:productPropertyTextual2 ?propertyTextual2 .
    %Product1% bsbm:productPropertyTextual3 ?propertyTextual3 .
    %Product1% bsbm:productPropertyNumeric1 ?propertyNumeric1 .
    %Product1% bsbm:productPropertyNumeric2 ?propertyNumeric2 .
    ?productFeature rdfs:label ?productFeatureLabel .
    ?productFeature rdfs:comment ?productFeatureComment.
    ?productFeature dc:publisher ?productFeaturePublisher.
    ?review bsbm:reviewFor  %Product1% .
    ?review rev:reviewer ?reviewer .
    ?review dc:title ?revTitle .
    ?reviewer foaf:name ?revName .
    ?reviewer bsbm:country ?country.
    ?reviewer dc:publisher ?reviewerPublisher.
    ?reviewer dc:date ?reviewerPublishDate.
}
```

B.2 LSLOD Queries

Listing B.14: Prefixes

```
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/>
PREFIX drugcategory: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/
    drugcategory/>
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bio2RDF: <http://bio2rdf.org/ns/bio2rdf#>
PREFIX purl: <http://purl.org/dc/elements/1.1/>
PREFIX kegg: <http://bio2rdf.org/ns/kegg#>
PREFIX diseasome: <http://www4.wiwiss.fu-berlin.de/diseasome/resource/diseasome
    />
PREFIX dailymed: <http://www4.wiwiss.fu-berlin.de/dailymed/resource/dailymed/>
PREFIX sider: <http://www4.wiwiss.fu-berlin.de/sider/resource/sider/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX linkedct: <http://data.linkedct.org/resource/>
```

Listing B.15: **S1**: Find all drugs along with their indications

```
SELECT ?genericName ?indication
WHERE {
    {
     ?da drugbank:genericName ?genericName.
     ?da  drugbank:indication  ?indication.
    }
    UNION  {
     ?da dailymed:name  ?genericName.
     ?da   dailymed:indication ?indication.
    }
}
```

Listing B.16: **S2**: Find all drug descriptions and chemical equations of reactions related to durgs from category Cathartics

```
SELECT ?drugDesc ?cpd ?equation
WHERE {
    ?drug drugbank:drugCategory drugcategory:cathartics.
    ?drug drugbank:keggCompoundId ?cpd .
    ?drug drugbank:description ?drugDesc .
    ?enzyme kegg:xSubstrate ?cpd .
    ?enzyme rdf:type kegg:Enzyme .
    ?reaction kegg:xEnzyme ?enzyme .
    ?reaction kegg:equation ?equation .
}
```

Listing B.17: **S3**: Find all drugs, together with the URL of the corresponding Web-pages as well as images

```
SELECT ?drug ?keggUrl ?chebiImage
WHERE {
    ?drug       rdf:type                  drugbank:drugs .
    ?drug       drugbank:keggCompoundId   ?keggDrug .
    ?keggDrug   bio2RDF:url               ?keggUrl  .
    ?drug       drugbank:genericName      ?drugBankName .
    ?chebiDrug  purl:title                ?drugBankName .
    ?chebiDrug  bio2RDF:image             ?chebiImage .
}
```

Listing B.18: **S4**: Find KEGG drug names of all drugs in DrugBank belonging to category Micronutrient

```
SELECT distinct ?drug ?title
WHERE {
    ?drug drugbank:drugCategory drugcategory:micronutrient .
    ?drug drugbank:casRegistryNumber ?id .
    ?keggDrug rdf:type kegg:Drug .
    ?keggDrug bio2RDF:xRef ?id .
    ?keggDrug purl:title ?title .
}
```

Listing B.19: **S5**: Find all drugs and their mass that affect humans and other mammals. For those having a description of their bioinformation, also return this description

```
SELECT ?drug ?transform ?mass
WHERE {
    ?drug drugbank:affectedOrganism 'Humans and other mammals' .
    ?drug drugbank:casRegistryNumber ?cas.
    ?keggDrug bio2RDF:xRef ?cas .
    ?keggDrug bio2RDF:mass ?mass .
    OPTIONAL { ?drug drugbank:biotransformation ?transform  }
}
```

Listing B.20: **S6**: Find diseases and corresponding drugs that target those diseases

```
SELECT ?drug ?disease ?name
WHERE {
    ?drug drugbank:molecularWeightAverage ?weight .
    ?drug drugbank:possibleDiseaseTarget ?disease .
    ?disease diseasome:name ?name .
}
```

Listing B.21: **S7**: Find drugs and their side effects with labels for the drug name "Sodium Phosphate" in dailymed

```
SELECT ?drug ?sidereffect ?label
WHERE {
    ?drugAlt sider:sideEffect ?sidereffect .
    ?sidereffect rdfs:label ?label .
    ?drug dailymed:name 'Sodium Phosphates' .
    ?drug owl:sameAs ?drugAlt .
}
```

Listing B.22: **S8**: Find diseases and corresponding drugs that target those diseases along with their labels

```
SELECT ?drug ?disease ?label
WHERE {
    ?disease diseasome:name ?diseasomename .
    ?disease drugbank:possibleDiseaseTarget ?drug .
    ?drug rdfs:label ?label .
}
```

Listing B.23: **S9**: Find intervention names and ids for the drugs in dailymed with drug dose, description, inactive ingredients as well as possible disease target

```
SELECT DISTINCT *
WHERE {
    ?intervention a linkedct:intervention.
    ?intervention linkedct:intervention_intervention_name ?intervention_name.
    ?intervention rdfs:seeAlso ?dailymedDrug .
    ?dailymedDrug dailymed:dosage ?dosage.
    ?dailymedDrug   dailymed:description ?description.
    ?dailymedDrug dailymed:inactiveIngredient ?inactiveIngredient .
    ?dailymedDrug dailymed:possibleDiseaseTarget ?possibleDiseaseTarget .
}
```

Listing B.24: **S10**: Find intervention names and types for the drugs in durgbank with drug chemical structure, drug state, its protein binding and smiles String Canonical

```
SELECT distinct *
WHERE
{
    ?intervention a linkedCT:intervention.
    ?intervention linkedCT:intervention_name ?intervention_name.
    ?intervention linkedCT:intervention_type ?intervention_type.
    ?intervention rdfs:seeAlso ?drugbankDrug.
    ?drugbankDrug drugbank:structure ?structure.
    ?drugbankDrug drugbank:state ?state.
    ?drugbankDrug drugbank:proteinBinding   ?proteinBinding.
    ?drugbankDrug drugbank:smilesStringCanonical ?smilesStringCanonical .
}
```

List of Figures

List of Tables