Semantic Data Integration and Knowledge Graph Creation at Scale

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur
(abgekürzt: Dr.-Ing.)

genehmigte Dissertation

von Frau
Samaneh Jozashoori, M.Sc.

2023

Referentin: Prof. Dr. Maria-Esther Vidal
Korreferentin: Prof. Dr. Ernestina Menasalvas
Tag der Promotion: 13.04.2023

Abstract

Contrary to data, knowledge is often abstract. Concrete knowledge can be achieved through the inclusion of semantics in the data models, highlighting the role of data integration. The massive growing number of data, in recent years, has promoted the demand for scaling up data management techniques; materializing data integration, a.k.a., knowledge graph creation falls in that category.

In this thesis, we investigate efficient methods and techniques for materializing data integration. We formalize the process of materializing data integration. We formally define the characteristics of a materialized data integration system that merge the data operators and sources. Owing to this formalism, both layers of data integration, including data and schema-level integration, are formalized in the context of *mapping assertions*. We explore optimization opportunities for improving the materialization of data integration systems. We recognize three angles including *intra/inter-mapping assertions* from which the materialization can be improved. Accordingly, we propose *source-based*, *mapping-based*, and *inter-mapping assertion* groups of optimization techniques. We utilize our proposed techniques in three real-world projects. We illustrate how applying these optimization techniques contribute to meeting the objectives of the mentioned projects.

Furthermore, we study the parameters impacting the performance of materialization of data integration. Relying on reported parameters and the presumably impacting parameters, we build four groups of testbeds. We empirically study the performances of these different testbeds in the presence and absence of our proposed techniques, in terms of execution time. We observe that the savings can be up to 75%.

Lastly, we contribute to facilitating the process of declarative data integration system definition. We propose two data operation function signatures in Function Ontology (FnO). The first set of functions is designed to perform the task of entity alignment by resorting to an entity and relation linking tool. The second library consists of domain-specific functions to align genomic entities by harmonizing their representations. Finally, we introduce a tool equipped with a user interface to facilitate the process of defining declarative mapping rules by allowing users to explore the data sources and unified schema while defining their correspondences.

Zusammenfassung

Im Gegensatz zu den Daten ist das Wissen oft abstrakt. Konkretes Wissen kann durch die Einbeziehung von Semantik in die Datenmodelle erreicht werden, was die Rolle der Datenintegration unterstreicht. Die massiv wachsende Zahl von Daten hat in den letzten Jahren die Nachfrage nach einer Ausweitung der Datenverwaltungstechniken gefördert; die materialisierende Datenintegration, auch bekannt als die Erstellung von Wissensgraphen, fällt in diese Kategorie.

In dieser Arbeit untersuchen wir effiziente Methoden und Techniken zur Materialisierung der Datenintegration. Wir formalisieren den Prozess der Materialisierung der Datenintegration. Wir definieren formal die Eigenschaften eines materialisierten Datenintegrationssystems, so dass die Datenoperatoren und -quellen zusammengeführt werden. Dank dieses Formalismus werden beide Ebenen der Datenintegration, einschließlich der Integration auf Daten- und Schemaebene, im Kontext von Mapping-Assertions formalisiert. Wir untersuchen die Optimierungsmöglichkeiten zur Verbesserung der Materialisierung von Datenintegrationssystemen. Wir erkennen drei Gesichtspunkte, einschließlich Intra-/Inter-Mapping-Assertions, unter denen die Materialisierung verbessert werden kann. Dementsprechend schlagen wir quellenbasierte, mappingbasierte und inter-mapping Assertionsgruppen von Optimierungstechniken vor. Wir setzen die von uns vorgeschlagenen Techniken in drei Forschungsprojekte ein. Wir veranschaulichen, wie die Anwendung dieser Optimierungstechniken dazu beiträgt, die Ziele der genannten Projekte zu erreichen.

Wir untersuchen die Parameter, die sich auf die Leistung der Materialisierung der Datenintegration auswirken. Auf der Grundlage der gemeldeten Parameter und der vermutlich ausschlaggebenden Parameter erstellen wir vier Gruppen von Testumgebungen. Wir untersuchen empirisch die Leistung dieser verschiedenen Testbeds mit und ohne die von uns vorgeschlagenen Techniken in Bezug auf die Ausführungszeit. Wir stellen fest, dass die Einsparungen bis zu 75% betragen können.

Schließlich tragen wir zur Erleichterung des Prozesses der deklarativen Definition von Datenintegrationssystemen bei, indem wir zwei Funktionssignaturen für Datenoperationen in der Function Ontology (FnO) vorschlagen. Die erste Gruppe von Funktionen ist für die Aufgabe des Entitätsabgleichs konzipiert, während die zweite Bibliothek aus domänenspezifischen Funktionen zum Abgleich genomischer Entitäten durch Harmonisierung ihrer Darstellungen besteht. Schließlich stellen wir ein Tool vor, das mit einer Benutzeroberfläche ausgestattet ist, um den Prozess der Definition deklarativer Mapping-Regeln zu erleichtern, indem es den Benutzern ermöglicht, die Datenquellen und das einheitliche Schema zu erkunden.

Preface

I have been a student since I remember; 25 years, to be more precise. Finishing each chapter of my study and following the next one has always come to me intuitively, without any hesitation, until reaching the Ph.D. chapter; it made me doubt my path many times and pushed me to the edge. Starting the second year of my Ph.D. I made a firm decision, I wanted to continue and conclude this chapter. The following four years of my study were full of adventures, activities, socialization, traveling, rejections, collaborations, emotions, burnout, bitter-sweet moments, and achievements! And today, I am writing my Ph.D. dissertation with a consent smile, thinking about all the people, without whom, I could not survive this journey.

I would like to first thank my supervisor, Prof. Dr. Maria-Esther Vidal, for giving me this opportunity, supporting me along the way, and showing patience during many hours of discussions that we had. She is a generous teacher and I learned a lot from her. I want to thank Prof. Dr. Sören Auer, for his great leadership. Although short, but our discussions during casual events were always fruitful and assuring for me. I also want to thank my colleague, Ahmad, who made me enjoy working on the projects once he joined the team. He and two other colleagues, Allard and Yaser were my close friends during the difficult time of the pandemic. Some days, the time that we were spending jogging together was the happiest time of my day. Also thanks to Enrique and David, my colleagues with whom I had most of my collaborations, hours of sharing thoughts, and interesting discussions. I would also like to thank all my past and present colleagues in SDM group, especially, Katja, Philip, Ariam, Kemele, Farah, Emetis, Mayra, and Hao.

I would like to thank my mother, Soheila, my all-time emotional support. In the darkest moments of my journey, she was the one who showed me the light and guided me through it. I also like to thank my father, Abbas, and my brother, Amir. They were always there for me when I needed them. They always made me believe that I "can". I want to thank my friend Eli, for supporting me and not giving up on me during the changes that I've been through in this journey. I also like to thank my uncle Mehrdad and aunt Tina, who always made me feel loved and supported while living so far from my parents.

Last but not least, I want to show my respect and gratitude toward all people who, no matter where or when or how, have fought for women's rights; those who paved the way and made it possible for me to stand where I stand today.

I dedicate this thesis to all the children around the world who are struggling with poverty and lack basic support to survive and thrive.

Contents

List of Figures

List of Tables

Acronyms

CSV	Comma Separated Values
CUI	Concept Unique Identifier from UMLS
DIS	Data Integration System
EA	Entity Alignment
ETL	Extract, Transform, Load
GAV	Global As View
JSON	JavaScript Object Notation
KG	Knowledge Graph
OWL	Web Ontology Language
R2RML	Relational to RDF Mapping Language
RDB	Relational Databases
RDF	Resource Description Framework
RDFS	RDF Schema
RML	RDF Mapping Language
UMLS	Unified Medical Language System
W3C	The World Wide Web Consortium

Part I

Preliminaries

Chapter 1

Introduction

Today, many publicly available data and knowledge bases provide the results of decades of research on different knowledge domains, targeting different use cases with different levels of reliability. Gaining a broad insight into each domain to derive accurate discovery and decision-making leans on transferring all these data into actionable knowledge through data integration. Integrating data residing in different sources demand the resolution of heterogeneity that may appear in the data. These heterogeneity conflicts can emerge in the schema of data or the data values that convey different representations of the same real world entities. Hence, data integration involves resolving the heterogeneity in data at the schema and data level.

There are two approaches in data integration including *materialization* and *virtualization* [8]. In the first approach, i.e., materialized data integration, data are transformed from different data sources into a centralized source. However, in the second approach, a virtual view of data residing in different sources is provided without moving and physically transforming data [78]. The updating process also differs between the two mentioned approaches. In the case of the first approach, once the data sources are updated, the process of materializing the data integration recurs. Considering the possible frequency of the demanded updates, the materialization process requires being equipped with scaled-up techniques.

The outcome of data integration can be encapsulated in graph-based structures named knowledge graphs (KG) [33]. Beyond the popularity that knowledge graphs have gained in research fields, they are critical in many enterprises [53]. Furthermore, knowledge graphs have promising potential in fields such as biomedicine. Considering the momentum that the idea of using semantics for data integration has gained, semantic data integration has imparted substantially to the knowledge graph creation infrastructures. This perspective is manifested by providing a unified view of data residing in different sources with heterogeneous structures. Consequently, the process of creating a knowledge graph can be defined based on materializing semantic data integration. In this thesis, we focus on *materialized data integration* and use this term and *knowledge graph creation* interchangeably.

1.1 Motivation and Challenges

The main motivation of this thesis originated in a real-world scenario where a materialized integration of massive heterogeneous biomedical data is required in order to improve the understanding of cancer diagnosis, prognosis, and treatment. Figure 1.1 motivates for heterogeneous data in different sources to be integrated and materialized. Ergo, this thesis focuses on the problem of integrating heterogeneous data at both schema and data levels and efficiently creating a knowledge graph. Despite the valuable scientific contributions, different challenges still exist on this topic. In the followings, the challenges tackled in this thesis are enumerated.

1.1.1 Challenge 1: A Transparent integration of data and metadata

A data integration system as a generic framework to define a knowledge graph is composed of a unified schema, a set of data sources, and mapping rules between concepts in the unified schema and the sources. By following the global as view (GAV) paradigm [47] where concepts in the unified schema are defined in terms of the sources, mapping rules enable the resolution of interoperability conflicts among data sources defined using different schemas. We call the resolution of this conflict, the **schema-level integration**. The mapping rules required to determine the correspondences between the data and the concepts of the unified schema can be defined utilizing declarative mapping languages such as R2RML [18] and RML [24]; it is a possible course of action to force traceability, which implies transparency.

In addition to the schema-level conflicts, data sources may have various levels of structuredness, suffer from various data quality issues, or expose different representations of the same real-world entities. Resolving the latest conflicts, a.k.a. **data-level integration**, is mostly performed by ad-hoc programs, preventing traceability and reproducibility. Nevertheless, the data-level integration can be defined transparently

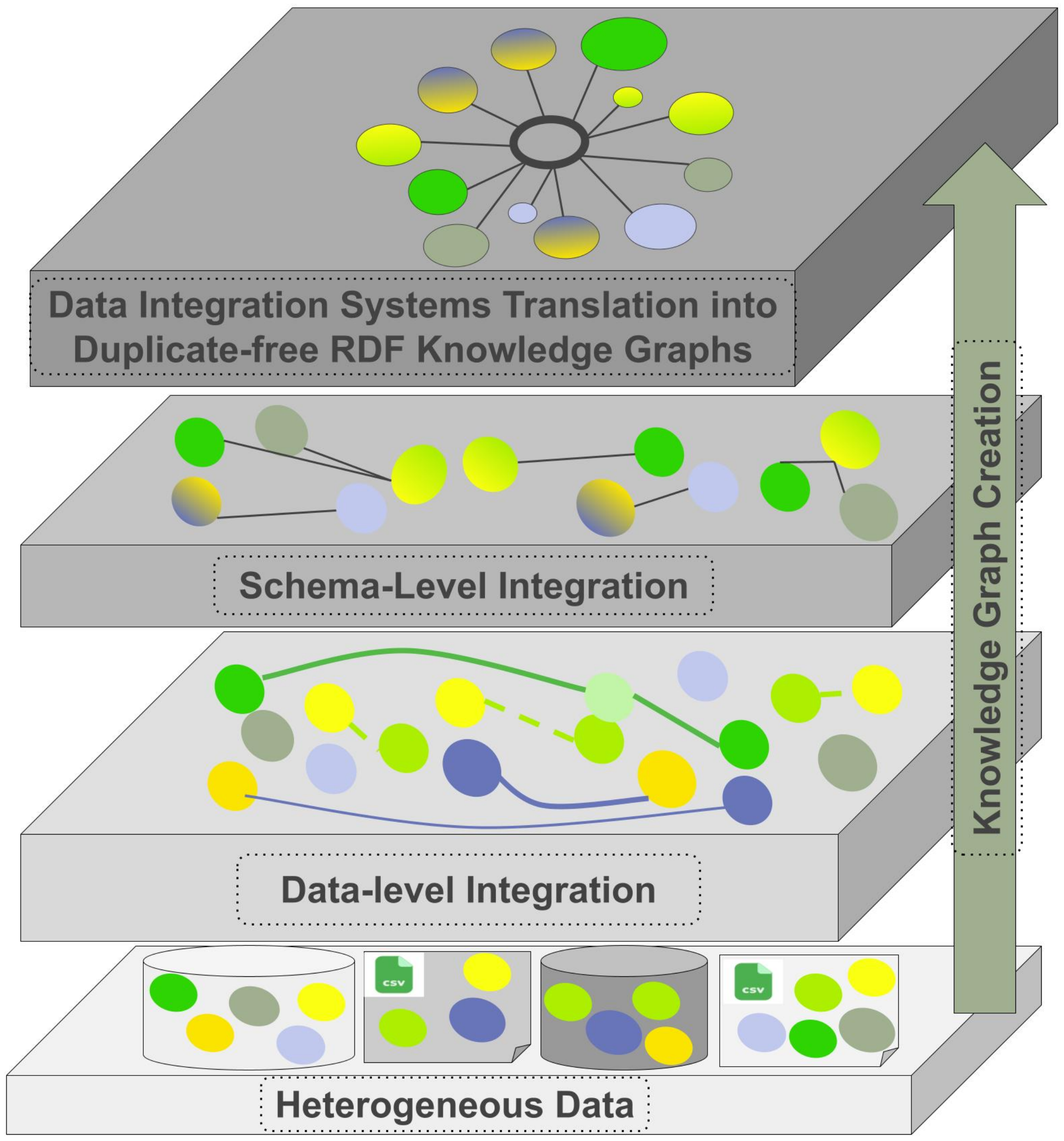


Figure 1.1: **Motivating Example.**.

by data operators [**abs-2105-0931**]. Data operators can be defined as functions that process and manipulate data which we call *data operation functions*. Owing to the existing formalism, data operation functions can be defined declaratively as part of the mapping rules [21, 44, 75, 19, 20]. The potential opportunities and challenges of applying data operation functions as part of the mapping rules to perform data-level integration tasks such as entity alignment are not well explored. Integrating and facilitating the transparent representation of specific data operation functions, such as entity alignment functions, is the first challenge that motivates this thesis and leads to the first two research questions.

1.1.2 Challenge 2: Efficiently Materializing data integration

RDF[1] or Resource Description Framework, as a standard model on the web for describing the metadata of resources, is a powerful data structure to represent factual statements created from heterogeneous data sources. In many domains, such as biomedicine and biology, a massive amount of generated data is unavailable in this format. Hence, knowledge graph creation demands being empowered by efficient data management techniques to be scaled up.

Additionally, albeit the advantages of integrating data operation functions in mappings, it introduces a new source of execution complexity that can negatively affect the efficiency of the translation and interpretation of the data integration system into RDF. Hence, data operation functions as a component of data integration systems need to be considered while optimizing and improving the efficiency of knowledge graph creation pipelines. Scaling up the process of knowledge graphs from given data integration systems considering all the components in an integrated fashion is the next challenge that we address in the context of the third research question.

1.1.3 Challenge 3: Identifying the Parameters negating the impact of the enhancement/optimization techniques

Enhancement techniques to speed up the knowledge graph creation need to be proposed in the context of parameters that impact the performance of knowledge graph creation pipelines. On the other hand, the influence of different parameters on the performance of knowledge graph creation pipelines is also subject to the applied components in the pipeline, e.g., data operation functions. A few studies evaluate the impact of different parameters considering different enhancement techniques [12, 3]. Nonetheless, there are important parameters that are yet to be studied or to be included in the existing benchmarks. Tackling this challenge brings us to another

[1]`https://www.w3.org/RDF/`

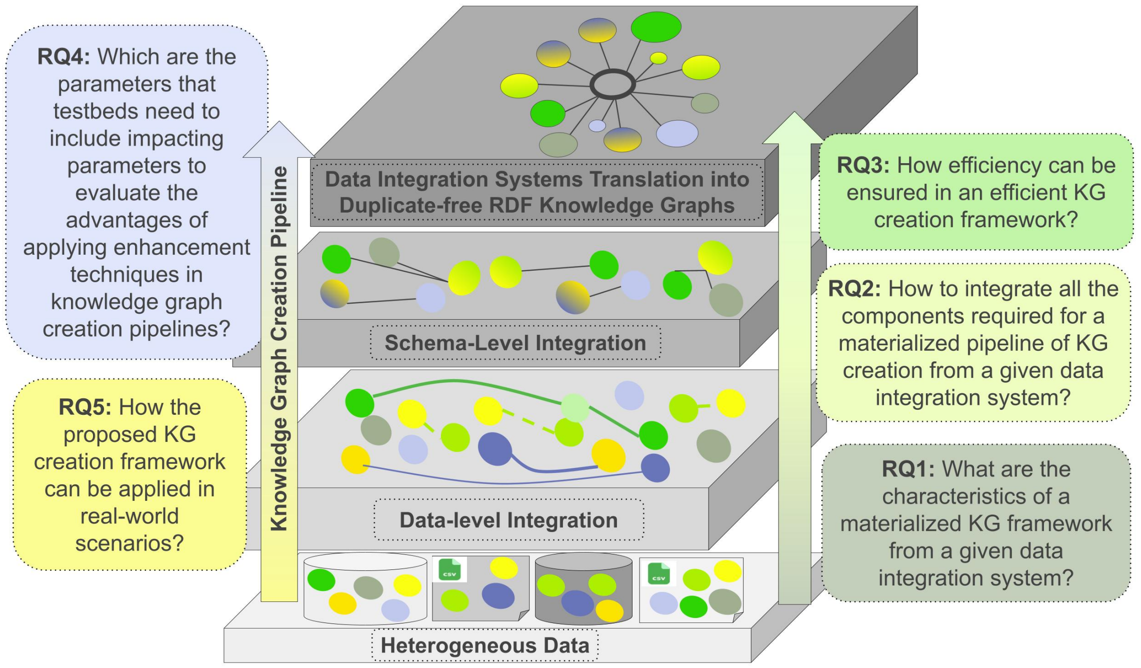

RQ4: Which are the parameters that testbeds need to include impacting parameters to evaluate the advantages of applying enhancement techniques in knowledge graph creation pipelines?
RQ5: How the proposed KG creation framework can be applied in real-world scenarios?
Knowledge Graph Creation Pipeline
Data Integration Systems Translation into Duplicate-free RDF Knowledge Graphs
Schema-Level Integration
Data-level Integration
Heterogeneous Data
RQ3: How efficiency can be ensured in an efficient KG creation framework?
RQ2: How to integrate all the components required for a materialized pipeline of KG creation from a given data integration system?
RQ1: What are the characteristics of a materialized KG framework from a given data integration system?

Figure 1.2: **The illustration of research questions**.

1.1.4 Challenge 4: Applying the materialized data integration in real-world scenarios

One of the questions arises after proposing a methodology or an approach is that how practical they are. Utilizing the proposed solutions in real-world scenarios and projects can illustrate their applicability and show their limitations in practice. This challenge motivates us to address our last research question.

1.2 Research Questions

Based on the challenges, we present the following research questions to be answered in this thesis.

RQ1: What are the characteristics of a materialized data integration system?

We study the KG creation process in terms of data integration systems. We define the components of the data integration system in such a way that it involves data-level integration in addition to schema-level integration.

RQ2: How to merge data operators and sources in a materialized data integration system?

With this question, we investigate the possibility of integrating data-level integration as part of the main framework of the knowledge graph. We aim to generate an integrated framework for knowledge graph creation.

RQ3: How can efficiency be ensured in a materialized data integration system?

Considering the components of a data integration system, we study the optimizations of knowledge graph creation from different perspectives. The data sources and the mapping assertions of a given data integration system can be transformed such that the transformed data integration system be more efficient in terms of knowledge graph creation. We propose different optimization techniques for improving the materialization of data integration systems.

RQ4: Which are the impacting parameters that testbeds need to include to evaluate the advantages of applying enhancement techniques in materialized data integration?

In addition to the known parameters impacting the performance of KG creation frameworks, we also consider parameters that we assume may impact the process of Kg creation. For this purpose, we provide different groups of testbeds that are configured to observe the impact of different parameters in the performance of knowledge graph creation frameworks that are applying different optimization techniques proposed in this thesis.

RQ5: How to overcome the challenges in applying materialized data integration systems in real-world scenarios?

Addressing this question, we propose a user interface platform to facilitate the creation of mapping rules, defining the correspondence between data sources and the concepts in the unified schema. Furthermore, we show the application of our proposed optimization techniques in different real-world projects building their knowledge graphs. First, we illustrate the importance of having scaled-up materialization of data integration systems in meeting the requirements of the exemplary projects. We show how different use cases with separate goals can benefit from our proposed optimization techniques.

1.3 Contributions

Tackling the problems represented by our research questions, we propose a set of optimization techniques Figure 1.3 for improving the performance of knowledge graph creation frameworks in terms of execution time. These optimization techniques improve the performance of the framework for a specific task while ensuring traceability. In the following, the main scientific contributions of this thesis addressing the five research questions are enumerated.

Contribution 1: Formalizing Materialized Data Integration Systems

The first contribution of this thesis includes the formalization of materialized data integration systems which also refers to knowledge graph creation. Relying on the existing formal definitions of data integration, we define the characteristics of materialized data integration systems,i.e., schema-level and data-level integrations. We investigate what components are required to be involved in data integration systems in order to carry on the defined characteristics. The components are recognized in terms of a unified schema, data sources, mapping rules, and functions. Moreover, we formally define two underrepresented components of the materialized data integration system, i.e., mapping rules and functions. This contribution addresses RQ1.

Contribution 2: Data-level Integration Solution

We explore the potential infrastructures for merging two important defined components of materialized data integration systems, i.e., mapping rules and functions. Owing to the formalisms provided by declarative mapping languages, we suggest applying these languages in defining data operation functions in the con-

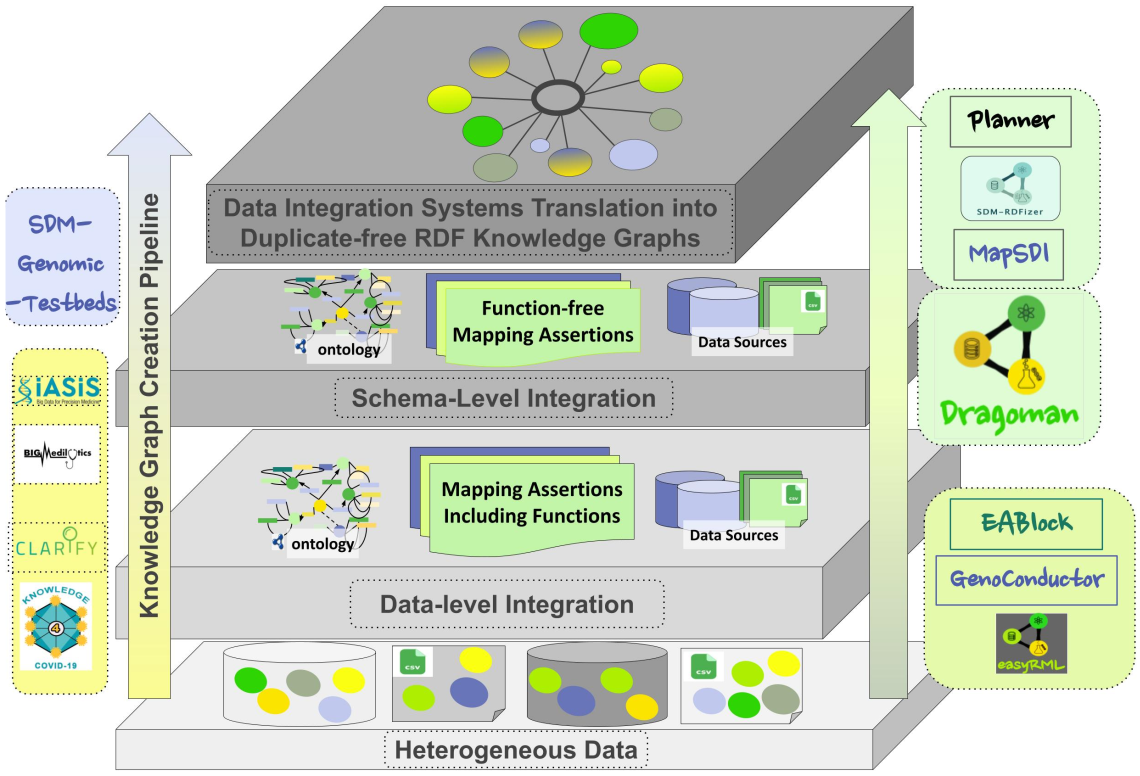


Figure 1.3: **Our Proposed Pipeline of Knowledge Graph Creation.**.

text of data-level integration. We show how applying mapping rules and their extensions enable traceability. This contribution addresses RQ1 and RQ2.

Contribution 3: Data Operation Function Libraries

Persuing the declarative data operation function definition, we provide two libraries of functions named *EABlock* and *GenoConductor*. *EABlock* is proposed to perform entity alignment tasks relying on the available state-of-the-art for solving the Named Entity Recognition (NER) and Entity Linking (EL) tasks. Relying on a tool performing entity alignment, *EABlock* links entities encoded in labels and short text to controlled vocabularies, i.e., UMLS and resources in encyclopedic, i.e., DBpedia and Wikidata, and other domain- specific knowledge graphs. *EABlock* functions are defined in a human and machine-readable medium, meeting meta-data requirements regarding transparency and reusability. *GenoConductor*, on the other hand, *GenoConductor*, provides a group of domain-specific functions linking genomic variation entities to a set of proposed harmonized representations; it solves genomic variant data heterogeneity. The functions in these libraries are defined using Function Ontology (FnO) [19], enabling them to be integrated as part of declarative mapping assertions defined by [R2]RML. This contribution target RQ2.

Contribution 4: Efficient Materialization of Data Integration System - Source-based Optimization

We address the efficient materialization of data integration systems considering different components of the system. We start with the data sources and improvements that can be performed from that angle. We introduce a framework, named *MapSDI*, consisting of a set of transformation rules at the level of data sources. The data integration system derived by applying the proposed transformation rules possesses two characteristics; **1.** the knowledge graph generated by this data integration system is *equivalent* to the knowledge graph obtained from the original data integration system and **2.** the execution time required to create the same knowledge graph from the transformed data integration system is *minimal*. This framework is one of the two main contributions of this author in tackling RQ3.

Contribution 5: Efficient Materialization of Data Integration System - Intra-mapping based Optimization

The second component to be considered is improving the mapping mapping rules. We propose a heuristic-based framework, *Dragoman*, specialized for optimizing the materialization of data integration systems in the presence of data operation functions. *Dragoman* is composed of a set of transformation rules to convert a given data integration system involving data operation functions into a function-free one. Relying on an eager evaluation strategy, first, *Dragoman* recognizes all the data operation functions in mapping rules and evaluates them. After the materialization of the functions, considering the mappings, *Dragoman* decides on the transformations required to transform an input data integration system into a function-free one. These transformations allow to transform the given data integration systems in such a way that the materialization is efficient and system-agnostic. Additionally, the transformation rules ensure the correctness and completeness of the transformed data integration system.

Contribution 6: Efficient Materialization of Data Integration System - Inter-mapping based Optimization

Another opportunity in improving the efficiency of data integration system materialization is planning the optimized execution of the mapping rules. To this end, we introduce *planner*, a heuristic-based approach. *Planner* provides an optimized execution plan by partitioning and scheduling the execution of the given mapping rules in the data integration system. This contribution serves to address RQ3.

Contribution 7: A Benchmark to Evaluate Materialized Data Integration Systems Incorporating the Significant Parameters

Considering the parameters that are reported [12] or questioned [43, 41, 36, 37] to impact the performance of materialized data integration systems, we produce a set of testbeds, named *SDM-Genomic-TestBeds*. These testbeds have been gradually created and extended along with the rest of the contributions of this thesis to assess the performance of each proposed framework or tool. The main focus of *SDM-Genomic-TestBeds* is targeting the parameters that are not gained attention in the existing benchmarks, such as GTFS-Madrid-Bench [13]. This contribution aims to address RQ4.

Contribution 8: Facilitating Mapping Assertion Creation

Committing to the application of [R2]RML and their extensions we provide a user interface named *easyRML* to facilitate the creation of mapping rules utilizing RML. Applying easyRML, users have the opportunity to traverse the unified schema and data sources while defining the mapping rules. This contribution serves to address RQ5.

Contribution9: Applying Efficient Materialization of Data Integration System in Real-world Projects

We show the requirements of knowledge graph creation frameworks of four real-world projects and how utilizing our proposed optimization techniques, these requirements can be met. These knowledge graphs are generated in the contexts of various EU projects including iASiS[a], BigMedylitics[b], CLARIFY[c], K4COVID[d]. This contribution addresses RQ5.

[a]https://project-iasis.eu/
[b]https://www.bigmedilytics.eu/
[c]http://www.clarify-project.eu/
[d]https://github.com/SDM-TIB/Knowledge4COVID-19

1.3.1 Publications

The majority of the contributions of this thesis are either already published or under review to be published in the contexts of the following peer-reviewed papers.

List of Published Peer-reviewed Papers

1. **Samaneh Jozashoori**, Maria-Esther Vidal. *MapSDI: A scaled-up semantic data integration framework for knowledge graph creation.* In Proceedings of the 27th International Conference on Cooperative Information Systems (CooPIS), 2019 Oct 21 (pp. 58-75). Springer, Cham. This paper is a joint work with my supervisor, Maria-Esther Vidal. My contributions are the definition of the approach, implementation, and evaluation of the proposed approach. I also set up the testbeds and configurations for the empirical study and performed them, explaining the results and visualizing them. ***Full research paper***

2. **Samaneh Jozashoori**, David Chaves-Fraga, Enrique Iglesias, Maria-Esther Vidal, and Oscar Corcho. *Funmap: Efficient execution of functional mappings for knowledge graph creation.* In International Semantic Web Conference, pp. 276-293. Springer, Cham, 2020. This paper is a collaboration with David Chaves-Fraga, a Ph.D. student at the Polytechnical University of Madrid, and our supervisors. My contribution is the motivation and definition of the approach. Furthermore, I defined the motivating examples, the testbeds for empirical studies, and the execution and analysis of the experimental studies. ***Full research paper***

3. **Samaneh Jozashoori**, Ahmad Sakor, Enrique Iglesias, and Maria-Esther Vidal. *EABlock: a declarative entity alignment block for knowledge graph creation pipelines.* In Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing, pp. 1908-1916. 2022. This paper is a joint work with Ahmad Sakor, a Ph.D. student at the Leibniz University of Hannover, and my supervisor, Maria-Esther Vidal. My contribution to this paper includes the creation of the motivating example, the definition of the approach, i.e., EABlock, the definition of functions, the implementation of the functions, defining the architecture of the proposed approach, the generation of the testbeds for the experimental studies, performing the experiments, and the visualization of the results. ***Full research paper***

4. **Samaneh Jozashoori**, Maria-Esther Vidal. *Data Integration for Supporting Biomedical Knowledge Graph Creation at Large-Scale.* In Proceedings of 13th International Conference, DILS 2018, Hannover, Germany, November 20-21, (2018). ***Short paper***

5. Enrique Iglesias*, **Samaneh Jozashoori***, David Chaves-Fraga*, Diego Collarana, and Maria-Esther Vidal. *SDM-RDFizer: An RML interpreter for efficiently creating RDF knowledge graphs.* In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, pp. 3039-3046. 2020. This article is a joint work with Enrique Iglesias, a research software developer, David Chaves-Fraga, a Ph.D. student at the Polytechnic University of Madrid, and my supervisor, Maria-Esther Vidal. The first three authors (shown by *) contributed equally to this publication. In this paper, my contributions include preparing motivating example, testbeds for experimental studies, and visualizing the results of the experiments. ***Full resource paper***

6. Enrique Iglesias, **Samaneh Jozashoori**, Maria-Esther Vidal. *Scaling Up Knowledge Graph Creation to Large and Heterogeneous Data Sources.* Journal of Web Semantics, (2022). This paper is a joint work with Enrique Iglesias, a research software developer, and my supervisor, Maria-Esther Vidal. In this paper, I contributed to the motivating example, formalizing mapping assertions, and testbeds for experimental studies. ***Full research paper***

7. Ahmad Sakor, **Samaneh Jozashoori**, Emetis Niazmand, Ariam Rivas, Kostantinos Bougiatiotis, Fotis Aisopos, Enrique Iglesias et al. *Knowledge4COVID-19: A Semantic-based Approach for Constructing a COVID-19 related Knowledge Graph from Various Sources and Analysing Treatments' Toxicities.* arXiv preprint arXiv:2206.07375 (2022). This article is a collaboration with a group of

Ph.D. students at the Leibniz University of Hannover, a group of researchers at Demokritos Institute of Greece, and my supervisor, Maria-Esther Vidal. In this paper, I contributed to creating the knowledge graph, including data cleaning, a unified schema creation, defining the required mapping rules using RML, and finally transferring the DIS into the knowledge graph. ***Full research paper***

8. Maria-Esther Vidal, **Samaneh Jozashoori**, and Ahmad Sakor. *Semantic data integration techniques for transforming big biomedical data into actionable knowledge.* In 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), pp. 563-566. IEEE, 2019. ***Short paper***

9. Maria-Esther Vidal, Kemele M. Endris, **Samaneh Jazashoori**, Ahmad Sakor, and Ariam Rivas. *Transforming heterogeneous data into knowledge for personalized treatments—A use case.* Datenbank-Spektrum 19, no. 2 (2019): 95-106. In this paper, I contributed to creating the knowledge graph including cleaning the data, creating a unified schema, understanding, cleaning, and semantifying data, and defining the required mapping rules using RML to integrate the data. ***Book Chapter***

10. Maria-Esther Vidal, Kemele M. Endris, **Samaneh Jozashoori**, Farah Karim, and Guillermo Palma. *Semantic data integration of big biomedical data for supporting personalized medicine.*. Current Trends in Semantic Web Technologies: Theory and Practice. Studies in Computational Intelligence, vol 815, pp. 25-56. Springer, Cham. (2019).In this paper, I contributed to the data integration and knowledge graph creation. ***Book Chapter***

11. Maria-Esther Vidal, Kemele M. Endris, **Samaneh Jozashoori**, and Guillermo Palma. *A Knowledge-Driven Pipeline for Transforming Big Data into Actionable Knowledge.* In International Conference on Data Integration in the Life Sciences, pp. 44-49. Springer, Cham, 2018. In this paper, I contributed to creating the knowledge graph pipeline, including cleaning the data, creating a unified schema, understanding, and semantifying data, defining the required mapping rules using RML to integrate the data, and finally transferring the data integration system into the knowledge graph efficiently. ***Short paper***

12. Calvo, V., E. Niazmand, E. Carcereny, **Samaneh Jozashoori**, D. Rodriguez, R. Lopez Castro, M. Guirado et al. *1730P Cancer long survivor artificial intelligence follow-up (CLARIFY): Family history of cancer and lung cancer.* Annals of Oncology 32 (2021): S1198-S1199. This article is a collaboration

with another Ph.D. student at the Leibniz University of Hannover, Emetis Niazmand, my supervisor Maria-Esther Vidal, and a group of clinicians in the EU project in that I am involved, CLARIFY. In this paper, I contributed to creating the knowledge graph, including cleaning the data, creating a unified schema for cancer patients, understanding the data and semantifying it, defining the required mapping rules using RML to integrate the data, defining and implementing the required FnO functions to integrate the data at the data-level, and finally transferring the DIS into the knowledge graph. ***Poster paper***

List of the Under-review Publications

1. **Samaneh Jozashoori**, Enrique Iglesias, and Maria-Esther Vidal. *Dragoman: Efficiently Evaluating Declarative Mapping Languages over Frameworks for Knowledge Graph Creation.* Semantic Web Journal (SWJ) (2022)[2]. This paper is a collaboration with Enrique Iglesias, a software researcher, and my supervisor, Maria-Esther Vidal. My contribution to this paper involves, defining the problem, providing the motivating example, formalizing the data operation functions in mapping assertions, defining the transformation rules, formalizing the transformed mapping assertions, defining the algorithm of the framework including the transformation rules, defining the architecture of the framework, defining the parameters and creating the testbeds required to evaluate the framework, performing the empirical studies, explaining the results of the experiments and visualizing them. ***Full research paper***

2. Fotis Aisopos, **Samaneh Jozashoori**, Emetis Niazmand, Disha Purohit, Ariam Rivas, Ahmad Sakor, Enrique Iglesias, Dimitrios Vogiatzis, Ernestina Menasalvas, Alejandro Rodriguez Gonzalez, Guillermo Vigueras, Daniel Gomez-Bravo, Maria Torrente, Roberto Hernández López, Mariano Provencio Pulla, Athanasios Dalianis, Anna Triantafillou, Georgios Paliouras, and Maria-Esther Vidal *Knowledge Graphs for Enhancing Transparency in Health Data Ecosystems.* Semantic Web Journal (SWJ) (2022)[3]. This article is a collaboration with a group of Ph.D. students at the Leibniz University of Hannover, my supervisor, Maria-Esther Vidal, and the partners of the EU project that I have been involved, named BigMedylitics. In this paper, I contributed to creating the knowledge graph including data cleaning, a unified schema creation, defining the required mapping rules using RML, and finally transferring the data integration system into the knowledge graph. ***Full research paper***

[2] `https://www.semantic-web-journal.net/content/dragoman-efficiently-evaluating-declarative-mapping-languages-over-framework`

[3] `https://www.semantic-web-journal.net/content/knowledge-graphs-enhancing-transparency-health-data-ecosystems-0`

3. Enrique Iglesias, Maria-Esther Vidal, **Samaneh Jozashoori**, Diego Collarana, and David Chaves-Fraga. *Empowering the SDM-RDFizer Tool for Scaling Up to Complex Knowledge Graph Creation Pipelines.*, Semantic Web Journal (SWJ), (2022)[4]. This paper is a joint work with Enrique Iglesias, a research software developer, and my supervisor, Maria-Esther Vidal. In this paper, I contributed to the testbeds and configurations for the experimental study. ***Full research paper***

1.4 Thesis Structure

The thesis is composed of four parts divided into 16 chapters, overall. Part I or preliminary part, provides an overview of the background knowledge in Chapter 2 and related works that are required to understand this thesis in Chapter 3. Part II involves six chapters, representing the proposed solutions to the problems tackled in the thesis. First, the components of a DIS are formally defined in Chapter 4. Then in the following two chapters, including Chapter 5 and Chapter 6 the contributions for the RQ1 are represented. Continuing with the Chapter 7 and Chapter 8, the data operation functions libraries and the user interfaces for creating mapping assertions as the solutions to RQ3 are explained. Last but not least, in this part, Chapter 9 explains the major contribution of this thesis, i.e., the Dragoman framework, in detail. Part III composed of five chapters, provides an overview of all empirical studies conducted in this thesis context. This part starts with Chapter 10 describing *SDM-Genomic-testbeds*, one of the main contributions of this thesis focusing on providing required benchmark evaluating knowledge graph creation pipelines. The evaluation part continues with four chapters consisting of Chapter 11, Chapter 12, Chapter 13, and Chapter 14 describing the details of the experimental setups and the results of the experiments on *MapSDI*, *Planner*, *EABlock*, and *Dragoman*, respectively. The last part, i.e., Part IV provides an insight into the applications of the main contributions of this dissertation, including different frameworks in Chapter 15. The last chapter of this part and the thesis, Chapter 16, summarizes all the chapters and concludes.

[4]https://www.semantic-web-journal.net/content/empowering-sdm-rdfizer-tool-scaling-complex-knowledge-graph-creatio

Chapter 2

Background

In this chapter, we explain the terminology and concepts that serve as the background knowledge of the the research presented in this thesis. In Section 2.1, we provide the definition of data integration system and the corresponding concepts. In Section 2.2, we explain the basic concepts in Semantic Web Technologies that serve as the foundation of our research. We follow this chapter with the description of knowledge graph creation in Section 2.3; it also includes the description of FAIR principles in Section 2.4.

2.1 Data Integration System

Data integration is the problem of providing a unified view of the heterogeneous data residing at separated sources [47]. Accordingly, a data integration system can be defined in terms of three components, i.e., O a unified schema or ontology, S a set of data sources, and M a set of assertions. In designing a data integration system, the main task is the definition of mappings representing the correspondence between the data at the sources and the unified schema. Two basic approaches for specifying the mapping in data integration systems are proposed including global-as-view (GAV) and local-as-view (LAV). The GAV approach follows the idea that the instances of each element of the unified schema need to be defined based on a view of given data sources. In contrast, the LAV approach relies on the idea that the values in each data source should be described in terms of a view of the unified schema.

Data integration involves two levels of integration. The first is the schema-level integration involves homogenizing different nomenclatures of the semantically same real-world concepts between schemas. The second is the data-level integration including harmonizing and aligning the identical data values representing the same real-world entity, represented differently in data sources [45]. Designing the unified schema and defining the correspondence between data and the concepts in the unified schema contribute to overcoming schema-level integration. Data-level integration requires identifying interoperability conflicts between data sources and providing solutions to them. Interoperability issues are explained as the following. ***Structuredness*:** this conflict occurs whenever data sources are described at different levels of structuredness, e.g., structured, semi-structured, and unstructured. ***Schematic*:** this interoperability conflict exists among data sources that are modeled using different schemas, e.g., different attributes representing the same concept. ***Domain*:** this conflict occurs among various representations of the same entity. They involve: i) homonym: the same name is used to exhibit concepts with different meanings, and ii) synonym: distinct names are used to model the same concept. Geisler et al. [31] introduce data operators for accessing, processing, or managing data in the data sources; we refer to them as *data operation functions* in this thesis. A data integration system, duly, can be defined as $DIS = \langle O, S, M \rangle$ without considering the data operation functions. Nevertheless, including data-level integration, we can define a data integration system in terms of four elements i.e., $DIS = \langle O, S, M, F \rangle$ where F represents a set of data operation functions to solve the data-level integration. The elements of DIS are described as follows:

- The global/unified schema O is defined as a triple, $O = (C, P, \mathit{Axioms})$ where C and P correspond to the signature of O and represent the classes and properties of O. The set *Axioms* denotes a collection of axioms staying the main characteristics of the properties of O; these asserted statements implicitly comprise knowledge describing the modeled universe of discourse.

- The data sources of DIS_G are represented by means of the set of signatures $S = \langle S_1^{Att_1}, \dots, S_n^{Att_n} \rangle$ where each symbol S_j stands for a data source, e.g., a file or relational table, and Att_j corresponds to the attributes of S_j.

- M is a set of mapping assertions defining the classes and properties in the unified schema O in terms of the sources in S. Mapping assertions enable the resolution of interoperability conflicts among data sources at the schema level.

- F is a set of functional symbols representing built-in and user-defined data operation functions. User-defined functions comprise any data operation functions such as entity alignment, data cleaning, and curation resolving the interoperability conflicts between data values in different sources.

2.2 Semantic Web

In 2001 Berners-Lee et al. stated that "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better-enabling computers and people to work in cooperation.*" The two main ideas of the Semantic Web and its technologies include, first, integrating and combining data residing in different sources according to a common format. The second idea is about a language to show the relation between the data and the real-world concepts and objects [5] in such a way that is understandable by a machine, in addition to a human. One of the most important contributions of the Semantic Web is the applicability of the Semantic Technology Stack on the data integration problem. Hence, Semantic Data Integration is applying semantic web technologies, i.e., the standards proposed by the World Wide Web Consortium (W3C), to solve data integration problems.

The Semantic Web provides formalisms for representing and accessing data that are translated to a set of standards and technologies used to create data stores, vocabularies, and write rules for handling data. At the core of these standards is the Resource Description Framework (RDF) and its associated schema languages, RDF Schema (RDFS), and the Web Ontology Language (OWL).

2.2.1 Resource Description Framework (RDF)

The Resource Description Framework (RDF) [77] is a graph-based data model to represent data and metadata on the Web. 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, a 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 such RDF triples is called an RDF graph, defined as a directed graph. Nodes in the RDF graph are resources or literals. RDF resources are identified by IRIs (Internationalized Resource Identifier) or blank nodes (anonymous resources or existential variables), while literals correspond to instances of a data type (e.g., numbers, strings, or dates). Semantically, an RDF triple states that a relationship, indicated by a predicate, holds between the resource denoted by the subject and the object. An IRI also identifies a predicate; it represents a binary property or relationship.

[5] `https://www.w3.org/2001/sw/`

2.2.2 RDF Schema (RDFS)

RDF Schema (RDFS) [9] is a semantic extension of RDF to describe the relationship between related resources. The RDF Schema (RDFS) [14] extends the RDF model with new predicates such as `rdfs:Class`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, and `rdfs:range`. The `rdfs:Class` allows for the specifications of a particular resource as a Class. Establishing a hierarchical structure between classes is realized using the `rdfs:subClassOf`. Relationships between classes are defined by properties, i.e., `rdf:Property`. Additionally, properties use `rdfs:domain` and `rdfs:range` to allow a more detailed specification of the relationships. The domain of a property specifies its subject type within a triple while, the range defines the type of an object. Applying `rdfs:subPropertyOf`, properties can be organized in a hierarchical fashion. RDFS enables the hierarchies according to the relationships defined by `rdfs:subClassOf` and `rdfs:subPropertyOf`.

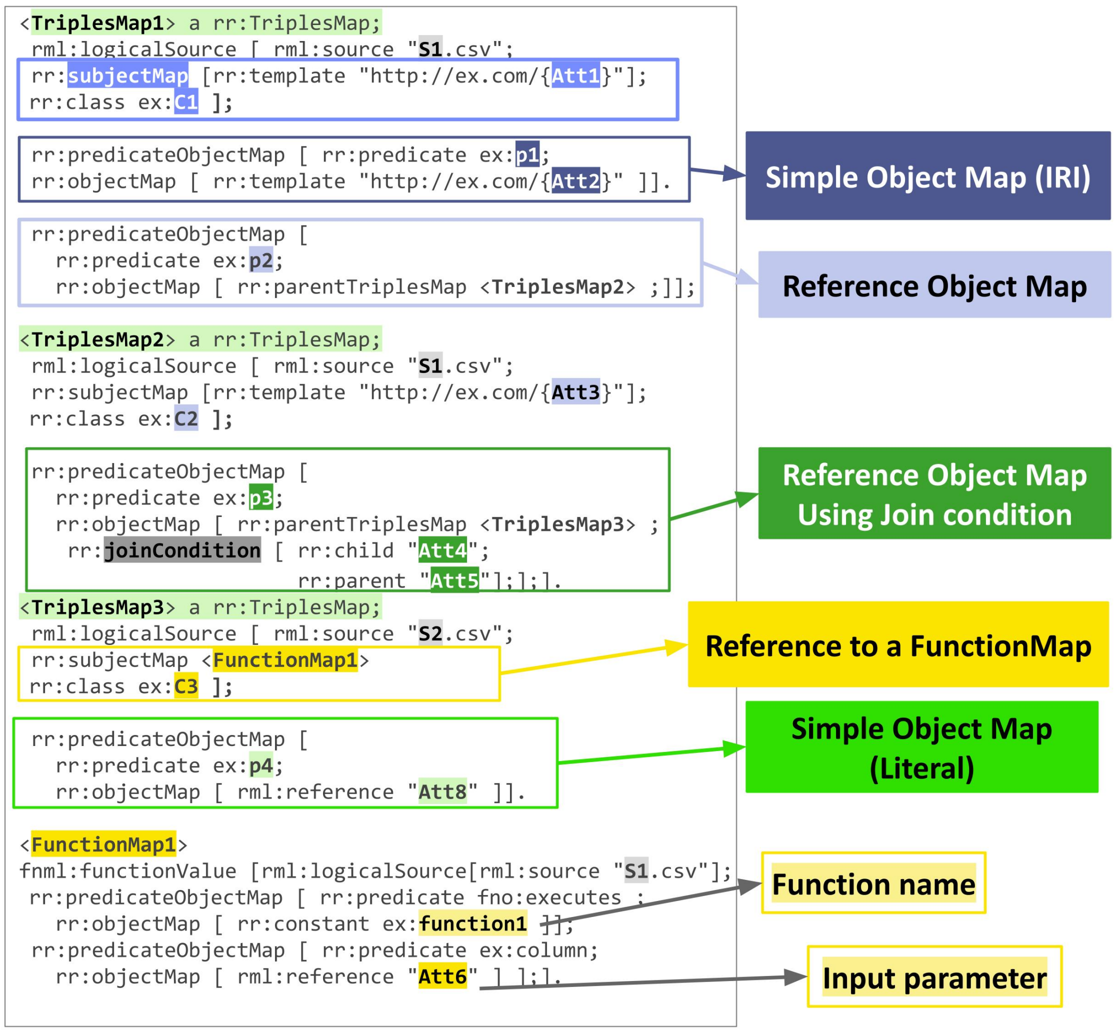
2.2.3 Web Ontology Language (OWL)

The Web Ontology Language (OWL) [49] is a Description Logics (DLs) based language introduced to assist in defining ontologies, i.e., a conceptualization of a domain knowledge [32] with explicit representation of the meaning of the terms and the relations between terms. OWL introduces new logical constructs and formal semantics at each level of the RDF model. To achieve the main objectives of the Semantic Web, introducing and applying an expressive language enabling the formal representation of terminologies and their relations is essential. In comparison to XML, RDF, and RDFS, OWL is considered to be a more expressive and powerful language which makes it the better choice for representing machine-interpretable content on the web.

2.2.4 Declarative Mapping Languages

Declarative mapping languages defined by the Semantic Web community can be used to define the correspondences between the data at the sources and the concepts in the unified schema. R2RML [18] is a language recommended by W3C to define mappings from relational databases (RDB)s to RDF datasets. RDF Mapping Language (RML) [23] is a declarative mapping language extending the W3C standard mapping language R2RML, used to transform heterogeneous data, including JSON, CSV, and XML, into the RDF data model. Each mapping assertion can be expressed as an RML mapping rule using its vocabulary[6]. Sim-

[6]`https://rml.io/specs/rml/`



```
<TriplesMap1> a rr:TriplesMap;
 rml:logicalSource [ rml:source "S1.csv";
 rr:subjectMap [rr:template "http://ex.com/{Att1}"];
 rr:class ex:C1 ];

 rr:predicateObjectMap [ rr:predicate ex:p1;
 rr:objectMap [ rr:template "http://ex.com/{Att2}" ]].

 rr:predicateObjectMap [
   rr:predicate ex:p2;
   rr:objectMap [ rr:parentTriplesMap <TriplesMap2> ;]];

<TriplesMap2> a rr:TriplesMap;
 rml:logicalSource [ rml:source "S1.csv";
 rr:subjectMap [rr:template "http://ex.com/{Att3}"];
 rr:class ex:C2 ];

 rr:predicateObjectMap [
   rr:predicate ex:p3;
   rr:objectMap [ rr:parentTriplesMap <TriplesMap3> ;
    rr:joinCondition [ rr:child "Att4";
                       rr:parent "Att5"];];].
<TriplesMap3> a rr:TriplesMap;
 rml:logicalSource [ rml:source "S2.csv";
 rr:subjectMap <FunctionMap1>
 rr:class ex:C3 ];

 rr:predicateObjectMap [
   rr:predicate ex:p4;
   rr:objectMap [ rml:reference "Att8" ]].

<FunctionMap1>
fnml:functionValue [rml:logicalSource[rml:source "S1.csv"];
 rr:predicateObjectMap [ rr:predicate fno:executes ;
   rr:objectMap [ rr:constant ex:function1 ]];
 rr:predicateObjectMap [ rr:predicate ex:column;
   rr:objectMap [ rml:reference "Att6" ] ];].
```

Simple Object Map (IRI)
Reference Object Map
Reference Object Map Using Join condition
Reference to a FunctionMap
Simple Object Map (Literal)
Function name
Input parameter

Figure 2.1: **RML Example**.

ilar to R2RML, a rule in RML is presented as a triples map (`rr:triplesMap`) defining the mapping between each entry in the data source to zero or more RDF triples, considering the provided ontology. Figure 2.1 illustrates an exemplary set of RML mapping rules. As it can be seen in Figure 2.1, rules are expressed by three `rr:triplesMaps`. A `rr:triplesMap` is defined according to the following characteristics. A `rr:triplesMap` must have exactly one logical source (`rml:logicalSource`) property, exactly one subject map (`rr:subjectMap`), and may have zero or more predicate-object maps (`rr:predicateObjectMap`) properties.

1. A `rml:logicalSource` is defined in terms of a `rml:source`, `rml:referenceFormulation`, and a `rml:iterator`. It should be noted that in the case of having a database or CSV as the `rml:logicalSource`, defining the `rml:iterator` is optional. In addition, the definition of `rml:referenceFormulation` is not required in the case of CSV, while, in the case of JSON or XML, all three properties are required to be defined.

2. RDF triples generated by each `Triples Map` share the same `SubjectMap`; shown in violet in Figure 2.1. A `SubjectMap` defines the resources of an RDF class in the unified schema.

3. A set of `rr:predicateObjectMap`s defines the properties and relations of a class specifying pairs of `rr:predicateMap`s and `objectMap`s which form RDF triples along with `rr:subjectMap`. The values of a `predicateObjectMap` can be defined in terms of a data source attribute, or as a reference or a join with the `rr:subjectMap` of another `triplesMap`. As shown in the example of Figure 2.1, a simple `rr:objectMap` can be created from the data attribute value as an IRI using `rr:template` (shown in **purple**) or as a literal using `rml:reference` (shown in green). A reference to another `triplesMap` is denoted as `rr:RefObjectMap`; it can be stated between `rr:triplesMap`s defined over the same data source (shown in light purple) or between `rr:triplesMap`s with *join dependency*. The *join dependency* between two `rr:triplesMap`s exist when the `rr:subjectMap` of one `rr:triplesMap` is the object of another `rr:triplesMap`. In this case, a *join* operator is required to be performed which is show applying a `rr:JoinCondition` (illustrated in **dark green**).

The Function Ontology (FnO) [19, 20] introduces a specification and ontology to semantically declare data operation functions. It allows for definitions of certain problems, executions, and implementations of the function into the bargain. Accordingly, RML is extended to integrate the declarative representation of data operations functions using FnO in the mapping rules, i.e., RML+FnO. Functions are defined by `functionMaps` in RML+FnO (appeared in yellow). A `functionMap` can be referenced by a `rr:subjectMap` or a `rr:predicateObjectMap`.

2.3 Knowledge Graph

Knowledge graphs are data structures that represent factual statements as entities and their relationships using a graph data model [33]. A KG is a directed graph G=(O,V,E), where: O is the unified schema, V is a set of nodes in the KG; nodes in V correspond to classes or instances of classes in O. E is a set of directed labeled edges in the KG that relate nodes in V. Edges are labeled with properties and relations in O. The foundation of any knowledge graph is built on the principles of modeling data as a graph [34]. We follow with the definition of the most common graph data models based on [34]:

- Directed Edge-labelled (del) Graphs are defined as a set of nodes, representing entities and a set of directed labeled edges between those nodes, representing binary relations between the entities. Adding new data to such a data model includes adding new nodes and edges. Accordingly, this data model offers more flexibility for integrating new data, compared to the standard relational model such as XML or JSON, where a schema must be defined upfront. An RDF graph is a W3C standard data model based on del graphs.

- In heterogeneous Graphs, each node and edge is assigned one type. Similar to del graphs, edge labels in heterogeneous graphs correspond to edge types. However, the types of nodes are part of the graph model, rather than representing a special relation. If the edge connects two nodes with the same type, the edge is called homogeneous; otherwise, it is called heterogeneous. Heterogeneous graphs allow for partitioning nodes based on their type, which makes them good candidates for tasks such as machine learning ones [35].

- Property Graphs allow a set of property–value pairs and a label to be associated with nodes and edges. In a del graph, an edge cannot be directly annotated, while, a property graph provides such a feature offering a more flexible data model. Property graphs are used in popular graph databases, such as Neo4j [2].

In this manuscript, we focus on RDF graph data model, a standard data model based on del graphs. In knowledge graphs expressed in the RDF, nodes can be resources or literals, and edges correspond to predicates. We refer to the RDF knowledge graph as the knowledge graph or KG.

2.3.1 Knowledge Graph Creation

The creation of a KG $\mathcal{G}$ is defined in terms of *evaluating* a data integration system $DIS_{\mathcal{G}} = \langle O, S, M, F \rangle$. Accordingly, the process or pipeline of KG creation involves the techniques and approaches for evaluating the provided data integration system. Considering RDF knowledge graphs, the pipeline of KG creation results in RDF triples created by evaluating the data integration system.

2.3.2 Evaluation of Functions

The evaluation of data operation functions can be derived using a lazy or eager evaluation approach. The evaluation of a function is called eager when the parameters of the function are evaluated before the function is executed [66]. Moreover, employing an eager evaluation strategy means that functions are executed “as soon as possible”. In contrast, a lazy evaluation approach leads to the evaluation of functions “as required”.

2.4 The FAIR Principles

The *digital objects* are referred as any published records including data sources, tools, and vocabularies. The Findable, Accessible, Interoperable, and Reusable (FAIR) principles are introduced to provide a guideline for characteristics that any digital object is required to hold in order to ensure their reusability [76]. The elements of the FAIR principles include as explained in [76] are as follow:

- Being *Findable*: **F1)** A globally unique and persistent identifier is designated to [meta]data. **F2)** Metadata is provided for data. **F3)** The identifier of data is involved in the metadata. Meta data are indexed in a searchable resource.
- Being *Accessible*: **A1)** [Meta]data can be retrieved by their unique identifier applying a standardized protocol; the protocol needs to meet the following requirements. the protocol is open, free, and universally implementable. The protocol enables necessary authentication and authorization procedures. **A2)** metadata are required to be accessible even though the data are no longer available.
- Being *Interoperable*:**I1)** A formal, accessible, shared, and broadly applicable language is utilized to represent [meta]data. **I2)** [meta]data are represented using vocabularies that follow the FAIR principles. **I3)** [Meta]data include valid references to other [meta]data.
- Being *Reusable*: **R1)** The accurate and relevant attributes of meta[data] are described. [Meta] data are: **R1.1)** released only with a clear and accessible data usage license, **R1.2)** associated with provenance, and **R1.3)** aligned with the domain-relevant community standards.

It should be noted that the FAIR principles highlight the concept of being “machine-actionable”. A digital object is considered to be “machine-actionable” when it provides increasingly more detailed information to an autonomously acting, computational data explorer. In addition to the digital object itself, being "machine-actionable" can also refer to the content of the contextual metadata surrounding the digital object.

Chapter 3

Related Work

This chapter reviews the state of the art, upon which, the works presented in this thesis are built. The works presented in Section 3.1 consist of the formalizations provided in data integration, techniques in integrating data, and approaches for creating knowledge graphs. In Section 3.1, we specifically focus on data-level integration techniques adopted in different approaches. Section 3.2 provides an overview of the contributions in developing declarative mapping languages for data integration. We follow this chapter with the approaches and engines introduced for materializing data integration systems and knowledge graph creation in Section 3.3. Lastly, in Section 3.4, we present the contributions to the evaluation of knowledge graph creation approaches.

3.1 Data Integration

Data integration is the problem of providing a unified view of the data residing in separated sources and the fundamental challenge in knowledge graph creation. Lenzerini [47] is one of the first, formalizing components of data integration systems. Lenzerini sets up a logical framework for *data integration* whose objective is to combine the data from different sources based on a *global schema*. He formalizes the concept of *mapping*; it is required to overcome the heterogeneity between the schemas of different data sources.

Years later, Doan et al. define the different types of heterogeneity that occur while integrating data [25]. They explain that heterogeneity can also appear at the data level and can be classified into two types. **a.** The first type occurs when the values in one source can be derived by a transformation function from the other sources. For instance, the values separated as two columns in one dataset may be concatenated and represented as one column in the other sources. Resolving such heterogeneity may require a deeper knowledge of the semantics of the data. Reconciling this kind of heterogeneity is possible by adding data operation functions to transform the data values. **b.** The second type of data-level heterogeneity is considered in cases where multiple representations of the same real-world entity are possible. The latest heterogeneity can be addressed and resolved by performing Entity Alignment (EA) techniques. The literature review in the following provides an overview of the contributions focusing on data-level integration in KG creation. Hence, integrating semi/unstructured data, e.g., texts, requires a semantic layer to resolve data-level heterogeneity; we call it *data-level integration*. Capiello et al. [10] include the data-level integration in a Data Ecosystem (DE) as a set of data operators including steps such as data cleaning, Named-Entity Recognition (NER), and Entity Linking (EL). Chessa et al. introduce [14] a methodology to add a semantic layer to a data lake and create a KG. Barroca et al. [5] extract metadata from textual descriptions and link them to entities in KGs utilizing NER and EL techniques, while Chu et al. propose a method to address the challenge of entity relations extraction [15]. Alternatively, mapping languages have been extended to embrace data operators as functions that can be included as programming scripts directly in the mapping rules which is explained in detail in Section 3.2.

3.1.1 Data-Level Integration

The methods and approaches proposed for data integration in the process of KG creation can be divided into two categories based. Both categories suggest ad-hoc programs for data-level integration, however, the first category emphasizes on resolving the data-level integration while generating the RDF entities as pre-processing. While, the second category of the approaches addresses the data-level integration after materializing the DIS and modeling the data into RDF triples, i.e., post-processing. We explain the state of the art in each category in the following.

Starting with the first category, Szekely, et al. [65] propose an approach for building knowledge graphs and devising the DIG system. This approach resorts to KARMA [45], a semantic DIS proposed by Gupta et al., for data integration at the level of schema. DIG system tackles a number of challenges including scalability. LIMES [52] proposed by Ngomo et al. is an efficient approach for link discovery in metric spaces. Relying on the mathematical characteristics of metric spaces, LIMES, can filter out a large number of instances that cannot meet the matching condition set by the user sufficiently. The common feature of all presented approaches in this category is that they consider any data operation, e.g., entity linking, as separate steps before materializing data integration systems. In other words, the data-level integration performed by these approaches is exhibited as pre-processing steps. Nonetheless, a drawback of developing pre-processing steps as *ad-hoc* programs for each specific knowledge graph pipeline is that they reduce traceability and maintainability at scale.

The second category consists of approaches focusing on automatic linked data integration and fusion. Collarana et al. [16] introduce MINTE, an integration framework that relies on the concept of RDF molecules to represent RDF entities semantically and be able to create, identify, and merge semantically equivalent RDF entities. LDIF introduced by Schultz et al. [58] is the framework that provides a set of independent tools supporting the process of interlinking RDF datasets. These tools can be divided into two groups. The first group includes data integration approaches focusing on linking tasks, e.g., Silk [39]. Silk can discover links between RDF resources based on the similarity between datatype properties considering link specifications provided by users. The second group performs the data fusion task. For instance, Sieve [50] proposed by Mendes et al. is a framework for quality assessment and fusion methods. Benbernou et al. [6] propose a semantic-based RDF data fusion relying on an inference mechanism using rules. Nevertheless, as mentioned earlier, the approaches explained in this category consider the data operations, such as entity alignment, as a differentiated step after the main integration process, a.k.a. post-processing. Hence, these approaches ignore the cost forced on the knowledge graph creation process due to generating the same nodes multiple times.

The explained limitations in both categories shed light on the necessity of integrating the data operation in the main process of materializing data integration and knowledge graph creation. Consequently, it is essential to introduce scaled-up approaches to build knowledge graphs from the data integration systems that involve data operation functions. Accordingly, we define the *data operation functions* as a component in data integration systems. This component serves as the foundation for data-level integration. Moreover, we propose optimization techniques to scale up the materialization of data integration systems involving data operation functions.

3.2 Declarative Definition of Data Integration

To specify the correspondences between the unified schema and the data sources transparently, a declarative mapping language can be applied. In 2012, the World

Wide Web Consortium (W3C) recommended R2RML [18], a language to declare mappings from relational databases (RDB)s to RDF datasets. In 2014, Dimou et al. introduce RDF Mapping Languages (RML) [24], the extension of R2RML. RML is a source-agnostic and extendable language supporting different source formats including XML and JSON, in addition to tabular data. These two exemplar mapping languages represent an infrastructure to resolve schema-level heterogeneity in a declarative and transparent manner. Accordingly, valuable efforts toward extending mentioned mapping languages have been made to merge the data operation function definitions in the mapping languages. Debruyne et al. introduce R2RML-F [21], an extension to R2RML to overcome interoperability issues while mapping the data in RDBs. De Meester et al. introduce Function Ontology (FnO) [19] to semantically declare data operation functions. Additionally, an extension of RML, FNML a.k.a RML+FnO, is provided to integrate the functions defined in FnO to the mappings declared in RML. Junior et al. propose another approach, FunUL [44], to merge the data operation functions in the mappings of CSV/Tabular data. In another attempt, Vu et al. introduce D-REPR [75], a new mapping language also involving data operations. Defining the data operation functions declaratively as part of the materialized data integration ensures the transparency of a knowledge graph creation framework. Furthermore, it enhances the data integration and transformation pipeline's maintainability, reusability, and reproducibility. Mentioned features lead to a materialized data integration accommodating FAIR guiding principles [76], which makes them a good replacement for data pre-processing steps.

3.3 Materializing Data Integration System and Knowledge Graph Creation

The semantic web community has contributed to proposing several methods and frameworks to materialize DISs with the mapping rules defined in R2RML and RML. In addition to the approaches explained in Section 3.1, engines are required to evaluate provided declarative DIS, materialize, and provide the materialized DIS in RDF data. These engines which are often referred to as *KG creation engines* can be compliant with different declarative mapping languages such as R2RML and RML. KG creation engines also demand to address the problem of scaled-up materialization [17]. Gawriljuk et al. [30] present a scalable framework for incremental KG creation which utilizes KARMA [45] for mapping the correspondence between data sources and the unified schema. On the other hand, Arenas-Guerrero et al. propose Morph-KGC [3], an RML-compliant engine that relies on an approach to partition

RML rules and execute them in parallel. Morph-KGC proposes an approach to partition R2RML and RML mapping assertions so that generated partitions can be executed in parallel. Morph-KGC relies on partitioning the mapping assertions into groups that generate disjoint sets of RDF triples. Nevertheless, based on this partitioning strategy, RDF triples with a join dependency, i.e., the subject of one RDF triple is the object of another, are partitioned into independent groups. Therefore, the same join RDF resource is generated redundantly by each disjoint partition to ensure the completeness and correctness of the result RDF triples. Nevertheless, an efficient partitioning strategy requires considering all mapping assertions including those that generate RDF triple sets with join dependency as a whole, to ensure that the result partitions are optimized. Therefore, despite the significance of all mentioned contributions and improvements, none of the mentioned approaches addresses the problem of scheduling the optimized execution of mapping assertion partitions, specifically considering different impacting factors, e.g., mapping assertions types, the connection between mapping assertions, and common properties among them. Additionally, the mentioned approaches are specific for an engine, i.e., they are not necessarily adaptable to generic KG creation pipelines.

Contrary to the engine translating data integration systems including RML mapping rules, engines able to translate RML+FnO have gained less attention and contribution. SDM-RDFizer [36], RMLMapper [22], RocketRML [63], and CARML[7] are the only accepted examples of the engines able to translate RML+FnO. Although valuable, these engines introduce no particular optimization for function execution; they follow a lazy evaluation strategy executing FnO functions. Additionally, an unpromising requirement for users to extend the available engines with their data operation functions is to get familiar with the already existing implementation of the engines. Users need to understand the implementation code of the engine to recognize where and how the implementation of new functions needs to be added.

All the mentioned drawbacks of available tools lead us to consider possible optimizations in both interpreting and evaluating complex mappings and functions while ensuring facilitated application and adaptation. We introduce engine-agnostic efficient execution techniques for different tasks of materialization. Our proposed frameworks optimize the evaluation of data operation functions and the transformation of data integration systems into function-free ones. Finally, they provide an optimized solution for materializing the transformed data integration systems relying on efficient partitioning and scheduling strategies. Any [R2]RML-compliant engine can adopt

[7]`https://github.com/carml/carml`

our proposed frameworks.

3.4 Benchmarking and Studies Reporting Parameters impacting Knowledge Graph Creation Performance

Namici et al. [51] compare two DIS materialization engines by formalizing the two systems considering W3C-compliant settings. In addition to the theoretical efforts, empirical evaluations such as the study by Chaves et al. [12] are conducted to define the parameters affecting DIS materialization and KG creation. Accordingly, benchmarks that consider the impacting parameters [12] are required to assess and compare the performance of different KG pipelines. One of the proposed benchmarks to evaluate different KG creation frameworks is GTFS-Madrid-Bench [13]; this benchmark provides a set of heterogeneous data and mappings. Despite the importance of considering reported variables, GTFS-Madrid-Bench lacks the requirements for studying all the impacting parameters reported in [12]. For instance, to evaluate the impact of data volume on different KG creation approaches, it is essential to have an equal growth of the volume in all the datasets involved in the KG; however, this requirement is not met by GTFS-Madrid-Bench. Furthermore, the deficiency of required testbeds to study parameters such as join selectivity, star-join, data duplicates, and duplicated predicates in mappings is another limitation of GTFS-Madrid-Bench. Therefore, to ensure the comprehensiveness of our experimental study, we create *SDM-Genomic-TestBeds*, a group of testbeds that consider all the reported parameters including those missing in the other benchmark, e.g., complex multi-sources role mapping assertions and percentage of duplicates.

There are no empirical studies reporting the parameters impacting the materialization of DIS in the presence of data operation functions. Accordingly, with a number of potential parameters being significant, we conduct a set of testbeds and experimental configurations to evaluate the materialization process including data operators. The results of these *preliminary* experimental studies, reported in Chapter 14, shed light on a number of influential parameters regarding the functions. According to these results, we extend *SDM-Genomic Testbeds* with specific testbeds considering the parameters that impact the materialization of DIS including functions in combination with the previously reported parameters that only target DIS materialization with no functions.

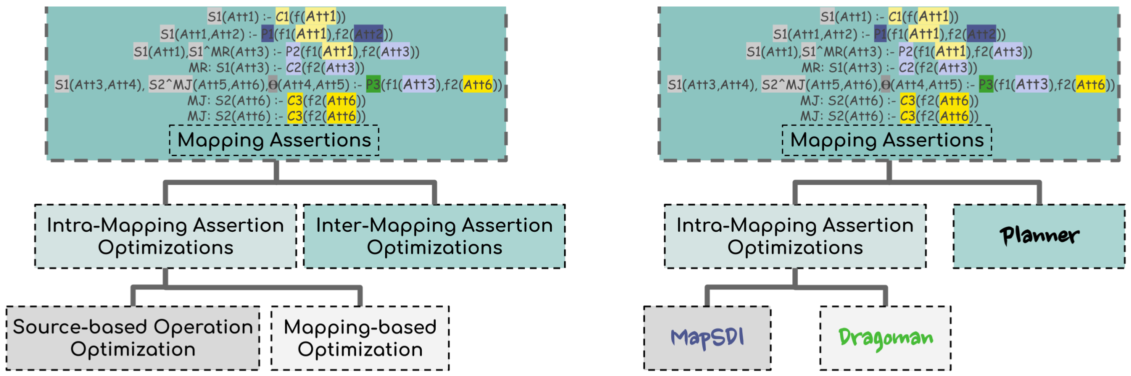
3.5 Summary

In this chapter we explain the state of the art in formalizing data integration, materializing data integration and knowledge graph creation, mapping languages, and corresponding benchmarks. We summarize the existing limitations of available approaches and potential opportunities for improvements in each area that we seek in this thesis.

Part II

Materialized Data Integration Systems

In this part, we present our proposed framework for knowledge graph creation. This framework mainly aims to improve the *efficiency* of materializing data integration systems.
Figure 3.1a shows an overview of the aspects that are targeted by this work at a glance. The main problem tackled in this part of the thesis can be summarized as follows. Given a set of mapping assertions as an element of the data integration system, the problem of creating a knowledge graph efficiently is defined as the problem of optimizing the evaluation of the mapping assertions. This optimization can be considered at two levels involving intra- and inter-mapping assertions; shown in Figure 3.1a. As the names suggest, intra-mapping assertion optimizations refer to the techniques that can be performed at the level of each individual mapping assertion, while, inter-mapping assertion optimizations concern the enhancement of a group of mapping assertions. Furthermore, the intra-mapping assertion optimization can be tackled at two levels; source- or mapping-based. Figure 3.1b shows the contribution of this thesis in different levels. The first contribution is an approach named *MapSDI* improving knowledge graph creation by proposing source-based enhancements. The second contribution, named *Dragoman*, focuses on the improvement of knowledge graph creation based on data integration systems including user-defined functions. Dragoman provides a set of mapping-based optimization techniques. *MapSDI* and *Dragoman* are the main contributions of this thesis which are extensively explained in Chapter 5 and Chapter 9 and evaluated in Chapter 11 and Chapter 14. Nevertheless, I also contributed to the motivation and evaluation of the *Planner*, a set of intra-mapping assertions optimizations to plan an efficient order of executing the mapping assertions which is briefly described in Chapter 6 and evaluated in Chapter 12.


S1(Att1) :- C1(f(Att1))
S1(Att1,Att2) :- P1(f1(Att1),f2(Att2))
S1(Att1),S1^MR(Att3) :- P2(f1(Att1),f2(Att3))
MR: S1(Att3) :- C2(f2(Att3))
S1(Att3,Att4), S2^MJ(Att5,Att6),θ(Att4,Att5) :- P3(f1(Att3),f2(Att6))
MJ: S2(Att6) :- C3(f2(Att6))
MJ: S2(Att6) :- C3(f2(Att6))
Mapping Assertions
Intra-Mapping Assertion Optimizations
Inter-Mapping Assertion Optimizations
Source-based Operation Optimization
Mapping-based Optimization
S1(Att1) :- C1(f(Att1))
S1(Att1,Att2) :- P1(f1(Att1),f2(Att2))
S1(Att1),S1^MR(Att3) :- P2(f1(Att1),f2(Att3))
MR: S1(Att3) :- C2(f2(Att3))
S1(Att3,Att4), S2^MJ(Att5,Att6),θ(Att4,Att5) :- P3(f1(Att3),f2(Att6))
MJ: S2(Att6) :- C3(f2(Att6))
MJ: S2(Att6) :- C3(f2(Att6))
Mapping Assertions
Intra-Mapping Assertion Optimizations
Planner
MapSDI
Dragoman

(a) The enhancement techniques for improving knowledge graph creation pipelines

(b) Contributions

Figure 3.1: The overview of the tackled problems and the solutions proposed in different layers.

Chapter 4

Formalizing Mapping Assertions

In this Chapter, we focus on the first challenge explained in Subsection 1.1.1 by providing the formalization of materialized data integration. We answer the following research questions in this chapter:

RQ1: What are the characteristics of a materialized data integration system?

RQ2: How to merge data operators and sources in a materialized data integration system?

Answering the mentioned research questions, we formalize the definition, semantics, and the interpretations of *mapping assertions*, an important component in data integration systems. In addition to answering RQ1, these formalization also target the RQ2 by including the formalization of the concept *data operation functions* to address data-level integration. Lastly, this chapter enlightens different strategies in evaluating the mapping assertions and data operation functions. The content of this chapter is partially published in [37] and partially under the review[8].

4.1 Mapping Assertions Formalization

A logic program describes the world with a finite set of facts, a set of assertions about pieces in the world, and a set of rules allowing one to deduce facts from the known facts [11]. Facts and rules can be represented as **Horn clauses**. Horn clauses with exactly one positive literal are *definite clauses* and can be of general

[8] `https://www.semantic-web-journal.net/content/dragoman-efficiently-evaluating-declarative-mapping-languages-over-f`

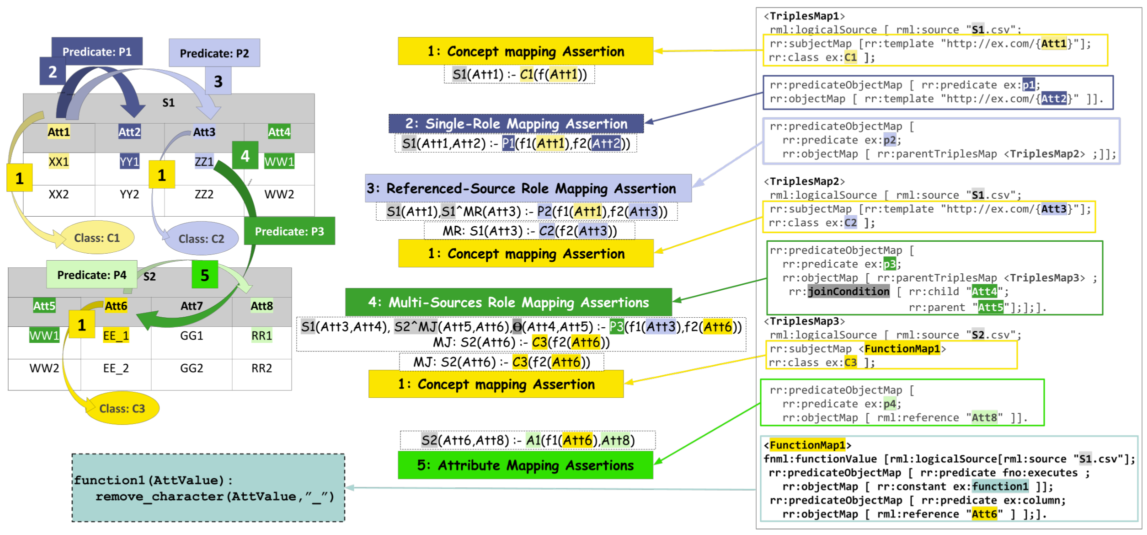
shape, $G_0 : -G_1, ..., G_n$ where the left-hand side of the clause is called its body and the right-hand side is the head; if the body holds, then the head holds: $body(\overline{X}) : -head(\overline{Y})$. $body(\overline{X})$ is a conjunction of predicates defined over terms, and $head(\overline{Y})$ is one predicate also defined over a set of *terms*. We define the *term* inductively as follows. **Base Case.** i) Let c be a constant. c is a term. ii) let X be a variable and X be a term. **Inductive Case.** Let h be a functional symbol of arity n and $t_1, ..., t_n$ be *term*s, then $h(t_1, ..., t_n)$ is a *term*.

4.1.1 Data Integration System and Knowledge Graph Creation

As explained in the Chapter 2, a data integration system can be defined as $DIS = \langle O, S, M, F \rangle$ where O represents the ontology that comprises classes, properties, and relations, S is a set of data sources, M represents a set of mapping assertions, and finally, F is a set of functional symbols representing built-in and user-defined functions. A knowledge graph is a directed graph generated from a data integration DIS defined as $KG = (O, V, E)$ where O represents the ontology, and V is a set of nodes in the KG; nodes in V correspond to classes or instances of classes in O. L is a set of directed labeled edges in the KG that relate nodes in V. Edges are labeled with properties and relations in O.

Considering the terminologies in R2RML [18] and RML [23] and its extension covering the application of FnO [19], a.k.a. FNML, user-defined functions in F represent the `fnml:FunctionTermMap`s which can express any data operation functions. However, built-in functions correspond to all predefined functions in [R2]RML such as `rr:template`. The function f in F can be considered "simple" or "composite". A simple function is a *term* $f(t_1, ..., t_n)$ where $t_1, ..., t_n$ are constant or variables. Contrary, function $f(t_1, ..., t_n)$ is composite, if any t_i is also a function.

Mapping rules or assertions in M are formalized as definite clauses, where $body(\overline{X})$ is a conjunction of predicates over the sources in S. The $head(\overline{Y})$ is defined as an n-ary predicate representing classes and properties in O, over a set of *terms*. In the following, we first provide the formal definition of different mapping assertions followed by examples illustrated in Figure 4.1. Lastly, the syntax of the RML mapping language representing different mapping assertions is explained with examples shown in Figure 4.1. The concept of mapping assertion extends the formalization presented in [37].


Predicate: P1
Predicate: P2
2
3
S1
Att1
Att2
Att3
Att4
XX1
YY1
ZZ1
WW1
XX2
YY2
ZZ2
WW2
1
1
4
Class: C1
Class: C2
Predicate: P3
Predicate: P4
S2
5
Att5
Att6
Att7
Att8
WW1
EE_1
GG1
RR1
WW2
EE_2
GG2
RR2
1
Class: C3
1: Concept mapping Assertion
S1(Att1) :- C1(f(Att1))
2: Single-Role Mapping Assertion
S1(Att1,Att2) :- P1(f1(Att1),f2(Att2))
3: Referenced-Source Role Mapping Assertion
S1(Att1),S1^MR(Att3) :- P2(f1(Att1),f2(Att3))
MR: S1(Att3) :- C2(f2(Att3))
1: Concept mapping Assertion
4: Multi-Sources Role Mapping Assertions
S1(Att3,Att4), S2^MJ(Att5,Att6),θ(Att4,Att5) :- P3(f1(Att3),f2(Att6))
MJ: S2(Att6) :- C3(f2(Att6))
MJ: S2(Att6) :- C3(f2(Att6))
1: Concept mapping Assertion
S2(Att6,Att8) :- A1(f1(Att6),Att8)
5: Attribute Mapping Assertions

```
function1(AttValue):
  remove_character(AttValue,"_")
```

```
<TriplesMap1>
 rml:logicalSource [ rml:source "S1.csv";
 rr:subjectMap [rr:template "http://ex.com/{Att1}"];
rr:class ex:C1 ];

 rr:predicateObjectMap [ rr:predicate ex:p1;
 rr:objectMap [ rr:template "http://ex.com/{Att2}" ]].

 rr:predicateObjectMap [
   rr:predicate ex:p2;
   rr:objectMap [ rr:parentTriplesMap <TriplesMap2> ;]];

<TriplesMap2>
 rml:logicalSource [ rml:source "S1.csv";
 rr:subjectMap [rr:template "http://ex.com/{Att3}"];
rr:class ex:C2 ];

 rr:predicateObjectMap [
   rr:predicate ex:p3;
   rr:objectMap [ rr:parentTriplesMap <TriplesMap3> ;
    rr:joinCondition [ rr:child "Att4";
                       rr:parent "Att5"];];].
<TriplesMap3>
 rml:logicalSource [ rml:source "S2.csv";
 rr:subjectMap <FunctionMap1>
rr:class ex:C3 ];

 rr:predicateObjectMap [
   rr:predicate ex:p4;
   rr:objectMap [ rml:reference "Att8" ]].

<FunctionMap1>
fnml:functionValue [rml:logicalSource[rml:source "S1.csv"];
 rr:predicateObjectMap [ rr:predicate fno:executes ;
   rr:objectMap [ rr:constant ex:function1 ]];
 rr:predicateObjectMap [ rr:predicate ex:column;
   rr:objectMap [ rml:reference "Att6" ] ];].
```

Figure 4.1: **Mapping Assertion Examples**.

Concept Mapping Assertions: define instances of classes C in an ontology O using the class predicate $C(.)$ over the results of the $f(.)$ receiving a *term* t as input arguments. The predicate $S(\overline{X})$ represents the conjunction of source signatures $S_1(\overline{X_1}), ..., S_k(\overline{X_k})$, and $\overline{X}$ corresponds to the set of all the variables in the mapping assertion m, i.e., $\overline{X}$ is the union of $\overline{X_1}, ..., \overline{X_k}$.

$$S(\overline{X}) : -C(f(t))$$

According to the [R2]RML terminologies, the concept mapping assertion corresponds to the `rr:subjectMap` where the attributes of the `rml:logicalSource` which represents data sources, define the instances of the classes in O. As illustrated with the example in Figure 4.1, $f(.)$ corresponds to a built-in function, represented by RDF predicate `rr:template` to enable the concatenation of strings, or user-defined functions, defined by `fnml:FunctionTermMap`. As shown in Figure 4.1, `fnml:FunctionTermMap` declares the definition of data operation functions (`function1`) over attributes of the `rml:logicalSource` (`Att6`).

Role Mapping Assertions: defines an instance of the properties in O between instances of two classes in O with the role predicate $P(.)$ over the data sources attributes. This mapping assertion is expressed by `predicateobjectMap` in [R2]RML. Role mapping assertion can be divided into three categories as follow.

Single-Role Mapping Assertions define $P(.,.)$ over the attributes of a single data source using $f_1(t_1), f_2(t_2)$ where f_1 and f_2 are in F and t_1 and t_2 are *terms*. An example of this mapping assertion is shown in Figure 4.1 in **purple**, where it defines the role predicate $P1(.,.)$ over the instances of the class `C1` using `Att1` in `S1` processed by the built-in functions $f1$, and the object values based on `Att2` in the same data source `S1` processed by the $f2$. Using the syntaxes of [R2]RML, single-role mapping assertion is expressed by `rr:objectMap`.

$$S(\overline{X}) : -P(\underbrace{f_1(t_1)}_{Cma}, f_2(t_2)) \quad \rightarrow \quad Cma: \; S_1(\overline{X}_{1,1}) : -C_k(f_1(t_1)))$$

Referenced-Source Role Mapping Assertions, similar to the previous assertion, a referenced-source role mapping assertion defines the instances of the properties in O between instances of two classes in O with values over the same conjunction of source signatures $S(\overline{X})$. However, this assertion allows defining the object over a term t_2, and a function, $f_2(.)$, which are utilized in another mapping assertion, M, to specify the instance of a class in O:

$$S_i(\overline{X}_{i,1}), S_i^{MR}(\overline{X}_{i,2}) : -P(f_1(t_1), f_2(t_2)) \quad , \quad MR: S_i(\overline{X}_{i,2}) : -C_j(f_2(t_2))$$

An example of referenced-source role mapping assertion is provided in violet where the predicate $P2$ is defined over the instances of the class `C1` using `Att1` and the instances of the class `C2` based on `Att3` which are characterized as another concept mapping assertion. In [R2]RML reference-source role mapping assertions are expressed by using `rr:parentTriplesMap` to reference the `objectMap` of one `triplesMap` (`TriplesMap1` in the Figure 4.1) to the `subjectMap` of another `triplesMap` (`TriplesMap2` in the Figure 4.1).

Multi-Sources Role Mapping Assertions: Contrary to the previous assertion, a multi-source role mapping assertion allows for expressing the instances of the properties in O between instances of two classes in O with values over two different sources. Since the sources S_i and S_j are different, a join condition.

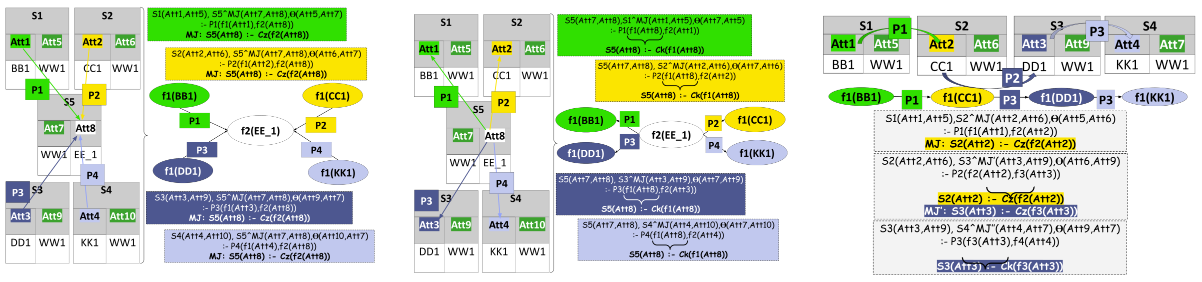
$$S_i(\overline{X}_{i,1}), S_j^{MJ}(\overline{X}_{i,2}), \theta(\overline{X}_{i,1}, \overline{X}_{i,2}) : -P(f_1(t_1), f_2(t_2)) \quad , \quad MJ: S_j(\overline{X}_{i,2}) : -C_z(f_2(t_2))$$

In Figure 4.1, an example of multi-sources role mapping assertion is shown in **dark green**. This mapping assertion defines the predicate `P3` over the instances of the class `C1` using `Att3` in data source `S1` and the instances of the class `C3` based on the values in `Att6` in the data source `S2`. The entries of two data sources `S1` and `S2` are connected by the join θ between the common fields in both data sources, i.e., `Att4` and `Att5`. Considering the [R2]RML language, similar to the referenced-source role mapping assertion, this assertion is expressed applying `rr:parentTriplesMap` connecting two `triplesMaps` (`TripelsMap1` and `TripelsMap2`). Additionally, the join condition between two `logicalSources` is defined by `rr:joinCondition` where `rr:child` represents the attribute in the `logicalSource` of the (`triplesMap`) referencing to another (`triplesMap`) as the object (`TriplesMap1` in this example). Also, `rr:parent` refers to the attribute in the `logicalSource` of the (`triplesMap`) that is referred to by `rr:parentTriplesMap` (`TriplesMap3` in Figure 4.1).

Attribute Mapping Assertions: defines the properties of a class in O using a predicate $A(.)$ over the values of attributes in terms of the conjunction of source signatures in $S(\overline{X})$.

$$S(\overline{X}) : -A(f(t_1), t_2)$$

In Figure 4.1 an example of attribute mapping assertion is presented (in green) which defines the predicate $A1$ over the instances of the class `C3` using `Att6` in `S1` and the literal data values in `Att8` in the same source. R2RML provides `rr:column` to represent attribute mapping assertions, while, RML introduces `rml:reference`.


S1
S2
Att1
Att5
Att2
Att6
BB1
WW1
CC1
WW1
P1
P2
S5
Att7
Att8
WW1
EE_1
P3
P4
S3
S4
Att3
Att9
Att4
Att10
DD1
WW1
KK1
WW1
S1(Att1,Att5), S5^MJ(Att7,Att8),Θ(Att5,Att7) :- P1(f1(Att1),f2(Att8))
MJ: S5(Att8) :- Cz(f2(Att8))
S2(Att2,Att6), S5^MJ(Att7,Att8),Θ(Att6,Att7) :- P2(f1(Att2),f2(Att8))
MJ: S5(Att8) :- Cz(f2(Att8))
f1(BB1)
f1(CC1)
f2(EE_1)
f1(DD1)
f1(KK1)
S3(Att3,Att9), S5^MJ(Att7,Att8),Θ(Att9,Att7) :- P3(f1(Att3),f2(Att8))
MJ: S5(Att8) :- Cz(f2(Att8))
S4(Att4,Att10), S5^MJ(Att7,Att8),Θ(Att10,Att7) :- P4(f1(Att4),f2(Att8))
MJ: S5(Att8) :- Cz(f2(Att8))
S5(Att7,Att8),S1^MJ(Att1,Att5),Θ(Att7,Att5) :- P1(f1(Att8),f2(Att1))
S5(Att8) :- Ck(f1(Att8))
S5(Att7,Att8), S2^MJ(Att2,Att6),Θ(Att7,Att6) :- P2(f1(Att8),f2(Att2))
S5(Att8) :- Ck(f1(Att8))
S5(Att7,Att8), S3^MJ(Att3,Att9),Θ(Att7,Att9) :- P3(f1(Att8),f2(Att3))
S5(Att8) :- Ck(f1(Att8))
S5(Att7,Att8), S4^MJ(Att4,Att10),Θ(Att7,Att10) :- P4(f1(Att8),f2(Att4))
S5(Att8) :- Ck(f1(Att8))
Att7
S1(Att1,Att5),S2^MJ(Att2,Att6),Θ(Att5,Att6) :- P1(f1(Att1),f2(Att2))
MJ: S2(Att2) :- Cz(f2(Att2))
S2(Att2,Att6), S3^MJ'(Att3,Att9),Θ(Att6,Att9) :- P2(f2(Att2),f3(Att3))
S2(Att2) :- Cz(f2(Att2))
MJ': S3(Att3) :- Cz(f3(Att3))
S3(Att3,Att9), S4^MJ''(Att4,Att7),Θ(Att9,Att7) :- P3(f3(Att3),f4(Att4))
S3(Att3) :- Ck(f3(Att3))

(a) Star Join (1) (b) Star Join (2) (c) Chain Join

Figure 4.2: **Examples of Different Combinations of Mapping Assertions.**

4.1.2 Combinations of Mapping Assertions

Given a group of mapping assertions. specifically the one including more than one multi-sources role mapping assertions, they can shape expensive forms in terms of execution. Here we define two of such forms called the "star join" and the "chain join". A group of multi-sources role mapping assertions form *star joins* if: **1.** more than one multi-sources mapping assertions include the same $MJ : S_j(\overline{X}_{i,2}) : -C_z(f_2(t_2))$, or **2.** more than one multi-sources role mapping assertions include the same concept mapping assertion $S_i(\overline{X}_{i,1}) : -C_k(f_1(t_1))$. To clarify, Figure 4.2a and Figure 4.2b each demonstrates one example of star joins corresponding to one of the mentioned conditions. As shown in Figure 4.2a, in this example the star join is created due to the fact that four separated multi-sources role mapping assertions have the same definition of $MJ : S_5(Att_8) : -C_z(f_2(Att_8))$, i.e., refer to the same object. However, the four different multi-sources role mapping assertions in Figure 4.2b generate a start join due to the fact that they all share the same concept mapping assertion, i.e., $S_5(Att_8) : -C_k(f_1(Att_8))$. Additionally, given a group of mapping assertions that is comprised of more than one multi-sources role mapping assertion, they generate a chain join if the definition of MJ in at least one of the multi-sources role mapping assertions is the same as the definition of another multi-sources mapping assertions. To better understand, Figure 4.2c depicts an example in which four different multi-sources role mapping assertions create a chain join. As it can be observed in Figure 4.2c, the definition of MJ in the first multi-sources role mapping assertion (the top one) is the same as the definition of concept mapping assertion in the second multi-sources role mapping assertion, i.e., $S_2(Att_2) : -C_z(f_2(Att_2))$. Furthermore, the definition of MJ' in the second multi-sources role mapping assertion is the same as the concept mapping assertion in the last multi-sources role mapping assertion, i.e., $S_3(Att_3) : -C_k(F_3(Att_3))$.

4.2 Semantics of Mapping Assertions

This section presents the formal semantics of the mapping assertion based on model theory. First, we define an interpretation structure of a data integration system $DIS = \langle O, S, M, F \rangle$, and the interpretation of the data sources in S in I, as well as the meaning of interpreting each of the mapping assertions of M in I.

Interpretation structure for a Data Integration System $DIS = \langle O, S, M, F \rangle$
Let $DIS = \langle O, S, M, F \rangle$ be a data integration system, an interpretation structure I for DIS is defined as follows: $I = \langle D, \sigma_c, \sigma_S, \sigma_F \rangle$, where

1. D is a non-empty set called domain of discourse
2. σ_c: for each constant c in O, S, M, and F, $\sigma_c(c)$ assigns a value in D for c
3. σ_S: for each source signature s in S, $\sigma_S(s)$ provides an interpretation of s in D. If s is an n-arity predicate, $\sigma_S(s)$ is a subset of D^n

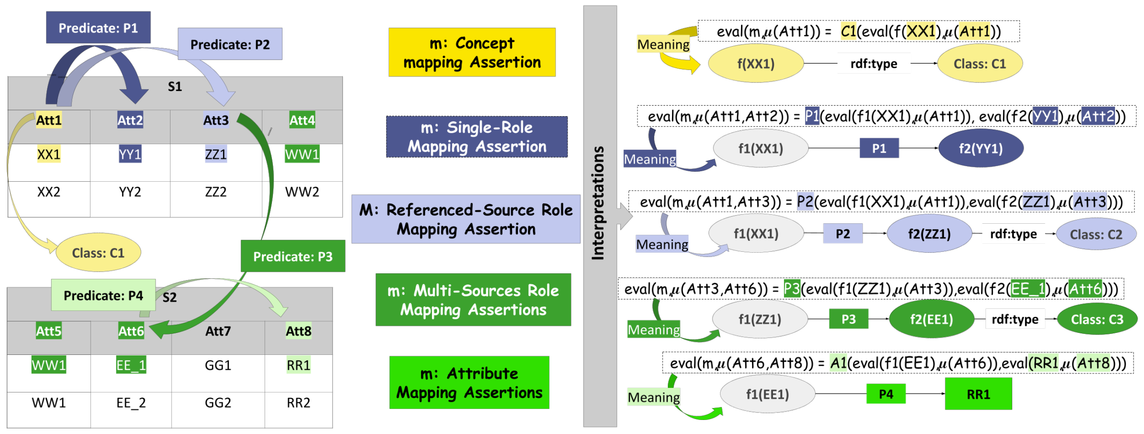

Predicate: P1
Predicate: P2
S1
Att1
Att2
Att3
Att4
XX1
YY1
ZZ1
WW1
XX2
YY2
ZZ2
WW2
Class: C1
Predicate: P3
Predicate: P4
S2
Att5
Att6
Att7
Att8
WW1
EE_1
GG1
RR1
WW1
EE_2
GG2
RR2
m: Concept mapping Assertion
m: Single-Role Mapping Assertion
M: Referenced-Source Role Mapping Assertion
m: Multi-Sources Role Mapping Assertions
m: Attribute Mapping Assertions
Interpretations
Meaning
eval(m,μ(Att1)) = C1(eval(f(XX1),μ(Att1))
f(XX1)
rdf:type
Class: C1
eval(m,μ(Att1,Att2)) = P1(eval(f1(XX1),μ(Att1)), eval(f2(YY1),μ(Att2))
f1(XX1)
P1
f2(YY1)
eval(m,μ(Att1,Att3)) = P2(eval(f1(XX1),μ(Att1)),eval(f2(ZZ1),μ(Att3)))
P2
f2(ZZ1)
Class: C2
eval(m,μ(Att3,Att6)) = P3(eval(f1(ZZ1),μ(Att3)),eval(f2(EE_1),μ(Att6)))
f1(ZZ1)
P3
f2(EE1)
Class: C3
eval(m,μ(Att6,Att8)) = A1(eval(f1(EE1),μ(Att6)),eval(RR1,μ(Att8)))
f1(EE1)
P4
RR1

Figure 4.3: **Mapping Assertion Interpretations**.

4. σ_F: for each functional symbol f in F, $\sigma_F(f)$ provides the interpretation of f in D. If f is an n-arity function, $\sigma_F(f) : D^n- > D$

Interpretation of Sources in a Data Integration System $DIS = \langle O, S, M, F\rangle$ Let S be a set of sources $S = \{S_1, S_2, ..., S_n\}$ in a data integration system $DIS = \langle O, S, M, F\rangle$. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F\rangle$ be an interpretation structure for DIS. The extension of the sources in S according to I, a.k.a. $E(S, I)$, is defined as follows:

1. $E : P^S \times \overline{I}- > P^{\overline{S}}$, where $\overline{I}$ is a set of interpretation structures of DIS; and $\overline{S}$ represents an infinite set of interpretations of source signatures from S according to interpretation structures in $\overline{I}$.

2. $E(S, I) = \{\sigma_S(S_1), \sigma_S(S_2), ..., \sigma_S(S_n)\}$, i.e., $E(S, I)$ returns for each of the source signature S_i in S, a set that corresponds to the interpretation of S_i in I.

Evaluation of Mapping Assertions: Interpretation of *terms* in mapping assertions. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F\rangle$ be an interpretation structure for DIS. Let t be a term in mapping assertion m in M. Let $\mu(\overline{X})$ be an assignment of the variables in t. The evaluation of the term t in $\mu(\overline{X})$ according to I, a.k.a. $eval(t, \mu(\overline{X}))$, is defined inductively as follows: **Base case:** a. The term t is a constant, $eval(t, \mu(\overline{X}) = \sigma_c(t)$ b. The term t is a variable and $\mu(\overline{X})|t$ is the value of t in $\mu(\overline{X})$, $eval(t, \mu(\overline{X}) = \sigma_c(\mu(\overline{X})|t)$. **Inductive case:** c. If h is an n-ary function and $t_1, t_2, ..., t_n$ are terms, and $t = h(t_1, t_2, ..., t_n)$. Then, $eval(h(t_1, t_2, ..., t_n), \mu(\overline{X}) = \sigma_F(eval(t_1, \mu(\overline{X}), eval(t_2, \mu(\overline{X}), ..., eval(t_n, \mu(\overline{X}))$
Interpretation of a Concept Mapping Assertion. Let $S(\overline{X}) : -C(f(t))$ be a concept mapping assertion in M, where $S(\overline{X})$ represents the conjunction of source

signatures $S_1(\overline{X_1}), ..., S_k(\overline{X_k})$, and $\overline{X}$ corresponds to the set of all the variables in the mapping assertion m, i.e., $\overline{X}$ is the union of $\overline{X_1}, ..., \overline{X_k}$. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F \rangle$ be an interpretation structure for DIS. Let $\mu(\overline{X})$ be an assignment of the variables in $\overline{X}$ to values in D, such that, each $S_i(\mu(\overline{X_i}))$ belongs to $\sigma_S(S_i)$ and the interpretation of $f(t)$ in $\mu(\overline{X})$, a.k.a. $eval(f(t), \mu(\overline{X}))$ belongs to $\sigma_F(f)$. Then, $eval(m, \mu(\overline{X})) = C(eval(f(t), \mu(\overline{X})))$. To simplify the perception, we illustrate the interpretation of a concept mapping assertion with an example shown in Figure 4.3 in yellow. As it is shown in Figure 4.3, given a concept mapping assertion as the m, the interpretation of m given `Att1`, i.e., $eval(m, \mu(Att1))$, provides an RDF triple representing an instance of the class `C1` in O with the value $f(Att1)$.

Interpretation of a Single-Role Mapping Assertion.

Let $S(\overline{X}) : -P(f_1(t_1), f_2(t_2))$ be a single-role mapping assertion in M, where $S(\overline{X})$ represents the conjunction of source signatures $S_1(\overline{X}_1), \ldots, S_k(\overline{X}_k)$, and $\overline{X}$ corresponds to the set of all the variables in the mapping assertion m, i.e., $\overline{X}$ is the union of $\overline{X}_1, ..., \overline{X}_k$. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F \rangle$ be an interpretation structure for DIS. Let $\mu(\overline{X})$ be an assignment of the variables in $\overline{X}$ the values in D, such that, each $S_i(\mu(\overline{X}_i))$ belongs to $\sigma_S(S)$ (for all i in {1,...,k}) and the interpretation of $f_1(t_1)$ and $f_2(t_2)$ in $\mu(\overline{X})$, a.k.a. $eval(f_1(t_1), \mu(\overline{X}))$ and $eval(f_2(t_2), \mu(\overline{X}))$ belong to $\sigma_F(f)$. Then, $eval(m, \mu(\overline{X})) = P(eval(f_1(t_1), \mu(\overline{X})), eval(f_2(t2), \mu(\overline{X})))$. In Figure 4.3, we demonstrate an example of the interpretation of a single-role mapping assertion given as m in **purple**. As it can be observed, the interpretation of m given `Att1` and `Att2` of the same data source `S1`, i.e., $eval(m, \mu(Att1, Att2))$, generates RDF triples representing the relation between the values of $f1(Att1)$ and $f2(Att2)$ defined by the predicate `P1` of O.

Interpretation of a Referenced-Source Mapping Assertion.

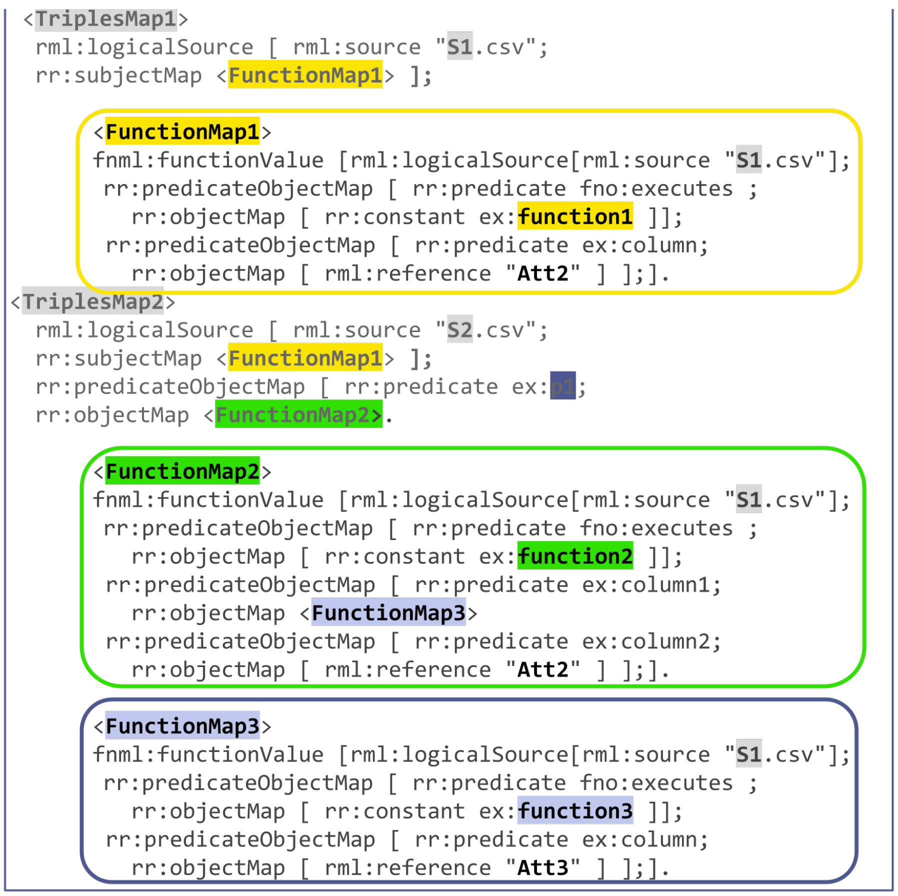
Let $S(\overline{X}_1), S^{MR}(\overline{X}_2) : -P(f_1(t_1), f_2(t_2))$ be a reference-role mapping assertion in M and MR is the referenced concept mapping assertion $S(\overline{X}_2) : -C(f(t))$. The predicates $S(\overline{X}_1)$ and $S^{MR}(\overline{X}_2)$ represent the conjunction of source signatures $S_1(\overline{X}_{1,1}), ..., S_k(\overline{X}_{1,k})$ and $S_1^{MR}(\overline{X}_{2,1}), ..., S_k(\overline{X}_{k,2})$, respectively. Let $\overline{X}$ be the union of the variables in $\overline{X}_1$ and $\overline{X}_2$. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F \rangle$ be an interpretation structure for DIS. Let $\mu(\overline{X})$ be an assignment of the variables in $\overline{X}$ in D, such that, each $S_i(\mu(\overline{X}))$ belongs to $\sigma_S(S_i)$, and the interpretation of $f_1(t_1)$ and $f_2(t_2)$ in $\mu(\overline{X})$, a.k.a. $eval(f_j(t_j), \mu(\overline{X})$ belongs to $\sigma_F(f)$. Then, $eval(m, \mu(\overline{X})) = P(eval(f_1(t_1), \mu(\overline{X}))$, $eval(f_2(t2), \mu(\overline{X})))$. To clarify, an example of the interpretation of m as a referenced-source mapping assertion is provided in Figure 4.3 in violet. The interpretation of m given `Attr1` and `Attr3` of the same source `S1`, generates RDF triples representing the relation between instances of two classes in O, i.e., `C1` and `C2` with the values in `Attr1` and `Attr3` respectively, considering the predicate `P2` in O.

Interpretation of a Multi-Sources Role Mapping Assertion.

Let $S(\overline{X}_1), S^{MJ}(\overline{X}_2) : -P(f_1(t_1), f_2(t_2))$ be a reference-role mapping assertion in M and MJ is the referenced concept mapping assertion $S^{MJ}(\overline{X}_2) : -C_z(f(t))$. The predicates $S(\overline{X}_1)$ and $S^{MJ}(\overline{X}_2)$ represent the conjunction of source signatures $S_1(\overline{X}_{1,1}), ..., S_k(\overline{X}_{1,k})$ and $S_1^{MJ}(\overline{X}_{2,1}), ..., S_t(\overline{X}_{t,2})$, respectively. Let $\overline{X}$ be the union of the variables in $\overline{X}_1$ and $\overline{X}_2$. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F \rangle$ be an interpretation structure for DIS. Let $\mu(\overline{X})$ be an assignment of the variables in $\overline{X}$ in D, such that, each $S_i(\mu(\overline{X}))$ belongs to $\sigma_S(S_i)$, each a $S_j^{MJ}(\mu(\overline{X}))$ belongs to $\sigma_S(S_j^{MJ})$, and the interpretation of $f_1(t_1)$ and $f_2(t_2)$ in $\mu(\overline{X})$, a.k.a. $eval(f_j(t_j), \mu(\overline{X})$ belongs to $\sigma_F(f)$. Then, $eval(m, \mu(\overline{X})) = P(eval(f_1(t_1), \mu(\overline{X}), eval(f_2(t2), \mu(\overline{X})))$. An example of the interpretation of m as a multi-sources role mapping assertion can be observed in Figure 4.3 in **dark green**. Similar to the interpretation of referenced-role mapping assertion, the interpretation of multi-sources role mapping assertion produce RDF triples representing the relation between instances of two classes in the O, i.e., `C1`and `C3`, using a predicate in the O, i.e., `P3` in this example. What differentiate the two interpretations are the data sources which contribute in the creation of the instances of `C1` and `C3`. As it can be seen in Figure 4.3, `Attr1` in the data source `S1` provides the instantiation of the class `C1`. However, the instances of the class `C3` are provided by the values of `Attr6` in the other data source, i.e., `S2`.

Interpretation of an Attribute Mapping Assertion.

Let $S(\overline{X}) : -A(f_1(t_1), f_2(t_2))$ be an attribute mapping assertion in M, where $S(\overline{X})$ represents the conjunction of source signatures $S_1(\overline{X}_1), \ldots, S_k(\overline{X}_k)$, and $\overline{X}$ corresponds to the set of all the variables in the mapping assertion m, i.e., $\overline{X}$ is the union of $\overline{X}_1, ..., \overline{X}_k$. Let $I = \langle D, \sigma_c, \sigma_S, \sigma_F \rangle$ be an interpretation structure for DIS. Let $\mu(\overline{X})$ be an assignment of the variables in $\overline{X}$ the values in D, such that, each $S_i(\mu(\overline{X}_i))$ belongs to $\sigma_S(S)$ (for all i in {1,...,k }) and the interpretation of $f_(t_1)$, a.k.a. $eval(f_1(t_1), \mu(\overline{X}))$, belongs to $\sigma_F(f)$ and the interpretation of t_2 in $\mu(\overline{X})$, a.k.a. $eval(t_2, \mu(\overline{X}))$, is true in I. Then, $eval(m, \mu(\overline{X})) = P(eval(f_1(t_1), \mu(\overline{X})), eval(t2, \mu(\overline{X})))$. The RDF triples generated by the interpretation of attribute mapping assertions represent the relation between instances of a class in O and a literal value extracted from an attribute in the same data source using a predicate in the O. In Figure 4.3 an example of the interpretation of an attribute mapping assertion is provided in **green**. It shows the relation between the instances of the class `C3` and literal values extracted from the `Att8` in `S2` using the predicate `P4`.



```
<TriplesMap1>
 rml:logicalSource [ rml:source "S1.csv";
 rr:subjectMap <FunctionMap1> ];

      <FunctionMap1>
      fnml:functionValue [rml:logicalSource[rml:source "S1.csv"];
       rr:predicateObjectMap [ rr:predicate fno:executes ;
         rr:objectMap [ rr:constant ex:function1 ]];
       rr:predicateObjectMap [ rr:predicate ex:column;
         rr:objectMap [ rml:reference "Att2" ] ];].
<TriplesMap2>
 rml:logicalSource [ rml:source "S2.csv";
 rr:subjectMap <FunctionMap1> ];
 rr:predicateObjectMap [ rr:predicate ex:p1;
 rr:objectMap <FunctionMap2>.

      <FunctionMap2>
      fnml:functionValue [rml:logicalSource[rml:source "S1.csv"];
       rr:predicateObjectMap [ rr:predicate fno:executes ;
         rr:objectMap [ rr:constant ex:function2 ]];
       rr:predicateObjectMap [ rr:predicate ex:column1;
         rr:objectMap <FunctionMap3>
       rr:predicateObjectMap [ rr:predicate ex:column2;
         rr:objectMap [ rml:reference "Att2" ] ];].

      <FunctionMap3>
      fnml:functionValue [rml:logicalSource[rml:source "S1.csv"];
       rr:predicateObjectMap [ rr:predicate fno:executes ;
         rr:objectMap [ rr:constant ex:function3 ]];
       rr:predicateObjectMap [ rr:predicate ex:column;
         rr:objectMap [ rml:reference "Att3" ] ];].
```

Figure 4.4: **Evaluation Strategies Example**. The two evaluation approaches including eager and lazy evaluations are shown for the same set of functions. Following an eager evaluation strategy, the function that is repeated in two mapping assertions, i.e., *function1* is evaluated only once. Also, *function3* which is an argument to *function2* is evaluated prior to *function2*. In contrast, following a lazy evaluation strategy, the evaluation of *function1* is repeated as many times as it is referred to in mapping assertions. As *function2* appears in mapping assertions prior to *function3*, a lazy evaluation strategy tries to evaluate *function2* prior to *function3* which cannot be completed and is revisited after the evaluation of *function3*.

4.3 Evaluation of User-defined Functions

The interpretation of $f(t)$ in $\mu(\overline{X})$, i.e., $eval(f(t), \mu(\overline{X}))$ can be derived using a lazy or an eager evaluation approach. The evaluation of a function is called eager when the parameters of the function are evaluated before the function is executed [66]. Moreover, employing an eager evaluation strategy means that functions, $f(t) \in F$, are executed "as soon as possible". In contrast, a lazy evaluation approach leads to the evaluation of functions "as required" [66]. In other words, following an eager evaluation strategy leads to evaluating all the defined functions in advance. Nevertheless, following a lazy evaluation means evaluating the functions once they are applied in mapping assertions and those mapping assertions are evaluated. For the sake of understandability, we explain the two strategies of the eager and lazy evaluation with the exemplar mapping assertions shown in Figure 4.4. The intuitive first step of an eager-evaluation-based approach is to traverse all the mapping assertions, find the ones including user-defined functions, and evaluate the functions. Accordingly, having the mapping assertions shown in Figure 4.4, such an eager-evaluation-based approach, first, detects four mapping assertions including user-defined functions. Then it starts evaluating them. However, since two of the functions are the same, the eager evaluation enables the approach to evaluate the duplicated function exactly once. Moreover, considering the fact that one of the arguments of the `function2` is the output of the `function3`, the eager evaluation forces the evaluation of the `function3` prior to the evaluation of the `function2`. Contrary, a lazy-evaluation-based approach starts executing the functions as soon as it reads the corresponding mapping assertions. A drawback of such an approach is that the duplicated functions are evaluated multiple times.

4.4 Summary

In this chapter, we formalize the definition and semantics of mapping assertions, an important component in data integration systems. These formalization serve as the foundations of the research we present in the following chapters.

Chapter 5

MapSDI: Inter-Mapping Assertion Optimizations

With the rapid growth of available big data in different domains, semantic data integration systems are required to be scaled up in order to transfer heterogeneous data into actionable knowledge represented in knowledge graphs. In many domains such as biomedicine and biology, a massive amount of generated heterogeneous data is required to be integrated to derive actionable knowledge. Therefor, materializing data integration and knowledge graph creation needs to be empowered with efficient processes for removing duplicates, projecting, and selecting relevant data attributes, and planning the mapping rules. In this chapter, Chapter 12, Chapter 7, and Chapter 9 we study the following research question and propose different solutions which can be, in fact, apply together.

RQ3: How can efficiency be ensured in a materialized data integration system?

We answer the mentioned research question, in this chapter, from the data source angle. As shown in the beginning of the Part II, addressing RQ3, we consider three possible aspects. The techniques proposed in this chapter ensures that the data sources in data integration systems are transformed in such a way that the required time for materialization is minimal. The contents of this chapter are based on three publications [43, 36, 37].

5.1 Motivating Examples

The motivation for the problem that we tackle in this chapter is illustrated in the Figure 5.1. This figure shows a traditional pipeline for transforming three

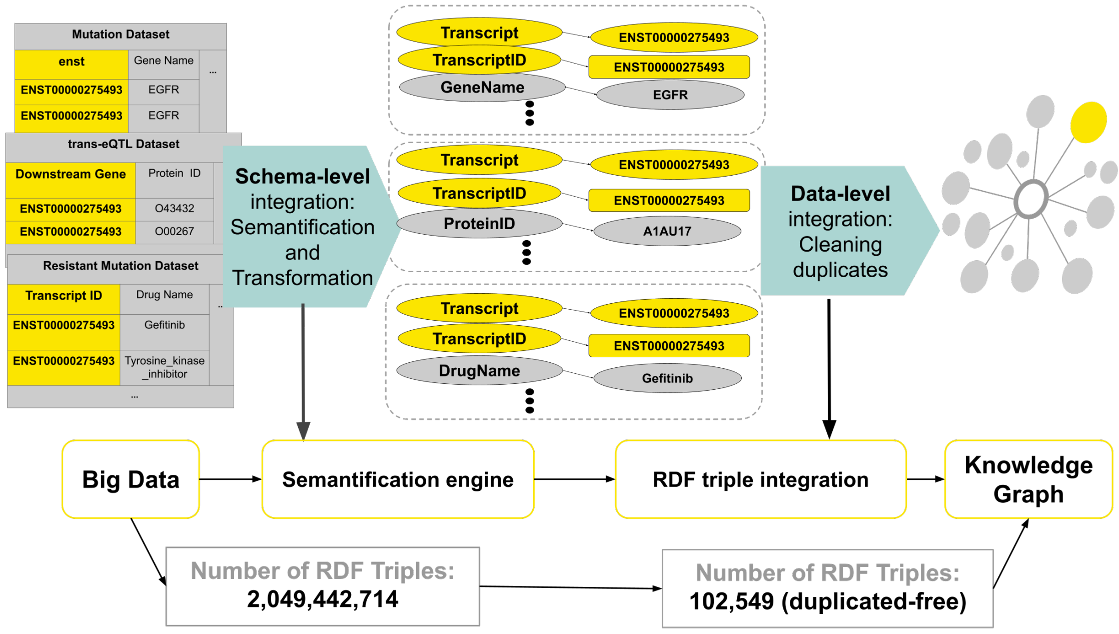


Figure 5.1: **Motivating Example**.

datasets into instances of a knowledge graph. The datasets contain information about mutations of genes, downstream genes, and drug resistance caused by mutations. These files are composed of up to 39 attributes (the mutation dataset), and their sizes are 186.4 MB, 71.9 GB, and 559 KB, respectively. The semantification of these datasets just for the concept "transcript" is performed using three RML triples maps, i.e., `triplesMaps`. These triples maps only consider the attribute in each dataset that represents the values corresponding to the concept "transcript". As we can see in Figure 5.1, the name of the attributes representing the concept "transcript" differ among the datasets; enst, downstream gene, and transcript id. This process ends up producing 2,049,442,714 RDF triples. However, because of overlaps across the three files, a large number of duplicates are generated, being reduced the output to only 102,549 duplicate-free RDF triples when cleaning and duplicate elimination are performed. Figure 5.1 illustrates this pipeline; it receives the three datasets and outputs the RDF triples to be included in the knowledge graph. As observed, in this real-world example, the pipeline for this semantic integration task is performed via two separated steps including: (I) **Schema-level integration**: Ontology-based data semantification and mapping rule-based data transformations. (II) **Data-level integration**: Redundancy elimination and cleaning. To explain the situation reported in this example, let us consider the meaning of these three datasets. A *transcript* refers to a ribonucleic acid via which a gene is expressed; it is used to synthesize a protein [48]. As it can be seen in Figure 5.1, *transcript* as a concept, can be represented with different labels in various databases which means that it cannot be distinguished and treated as the same concept unless being semantified according to the unified schema. Therefore, the first step of integration in the framework is to unify all the concept representations residing in different datasets by defining RML triple maps while transforming the data into RDF. The data semantification allows for also detecting duplicated data that were not recognizable before. Consequently, in the second step, the redundant data that are now represented as RDF triples are eliminated. It should be noted that the overall number of generated triples from different sources is 16,445 times the number of non-redundant triples which means that there is a considerable amount of duplicated data that could not be detected in the raw files. Considering the fact that similarity-based comparisons between RDF triples are more expensive than between the relational data model, specifically in the case of having a huge amount of data, leaves room to think about providing a more efficient and low-cost approach to create knowledge graphs. In this paper, we address the problem of semantic data integration motivated in this example, and present jozashoori2019mapsdi,I, a framework able to pre-process input datasets and avoid the generation of duplicated RDF triples. MapSDI is able to extract from the RML triple maps the knowledge required to pre-process the input datasets by means of the execution of basis relational algebra operations like the projection of attributes. Albeit simple, the transformations executed by MapSDI enable to project out only attributes that are utilized in the three triple maps, allowing the RDFizer to produce 102,549 duplicate-free RDF triples.

5.2 MapSDI Framework

Given a data integration system $DIS_G = \langle O, S, M\rangle$, the *problem of knowledge graph creation* is defined as the problem of identifying a data integration system $DIS'_G = \langle O, S', M'\rangle$ such that:

- The results of evaluating the two data integration systems is the same, i.e., *RDFize*$(DIS_G = \langle O, S, M\rangle)$=*RDFize*$(DIS'_G = \langle O, S', M'\rangle)$.

- The execution time of the evaluation of *RDFize*$(DIS'_G = \langle O, S', M'\rangle)$ is *minimal*, i.e., there is no other DIS''_G different from DIS'_G that generates the same RDF knowledge graph G but in a lower execution time.

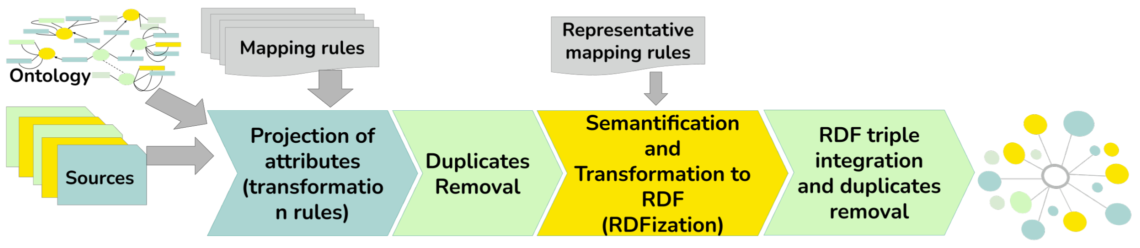

Ontology
Mapping rules
Representative mapping rules
Sources
Projection of attributes (transformation rules)
Duplicates Removal
Semantification and Transformation to RDF (RDFization)
RDF triple integration and duplicates removal

Figure 5.2: **MapSDI Framework**.

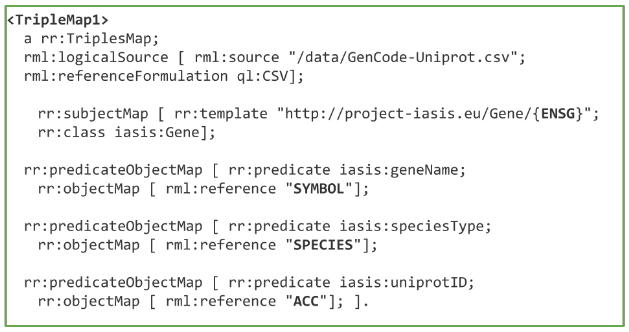
Proposed Solution: We propose MapSDI, an optimized alternative to traditional semantic data integration pipelines to create knowledge graphs. As it is shown in Figure 5.2, MapSDI receives a data integration system $DIS_G = \langle O, S, M \rangle$ as input and generates an RDF knowledge graph that corresponds to the result of evaluating $Eval(DIS_G = \langle O, S, M \rangle)$. Without lost of generality, MapSDI assumes that the mapping assertions in M are represented in a mapping language, e.g., the RDF mapping language RML. Before evaluating the function *Eval(.)*, MapSDI applies transformations to the sources in S and the mapping assertions in M in order to generate a data integration system $DIS'_G = \langle O, S', M' \rangle$ that corresponds to a solution of the *problem of knowledge graph creation*. MapSDI resorts to transformation rules applied to mapping assertions and sources depending on the attributes, variables, and sources that compose the mapping assertions in M. In other words, in a mapping assertion ma_i, the attributes from the data sources in the *Body* of ma_i are detected, and the corresponding sources in S are transformed to S' such that the data sources in the S' include only the attributes utilized in the mapping assertions. Accordingly, mapping assertions are also rewritten with the aim of reusing the attributes of the sources in S'. By projecting out only the attributes required in the head of mapping assertions, duplicates from the extensions of the sources are removed, avoiding thus, the generation of the same RDF triple multiple times during the evaluation of the function *Eval(.)*. Since only duplicates in the data sources are removed from the input, the resulting knowledge graph remains the same, while the time of producing duplicated RDF triples is reduced.

5.2.1 Transformation Rules

We present the transformation rules applied by the MapSDI framework in order to reduce duplicated data and speed up the execution time of the evaluation of a data integration system. The transformation rules are based on the axioms of the relational algebra [69] and in particular, the ones that stay when the project operator can be pushed down into the relations in a relational algebra expression. Furthermore, MapSDI extracts information from the mapping rules to decide when two or more datasets have equivalent attributes while represented with different attribute labels and must be merged into one file; and in case the merging is conducted, the corresponding rules are also merged.

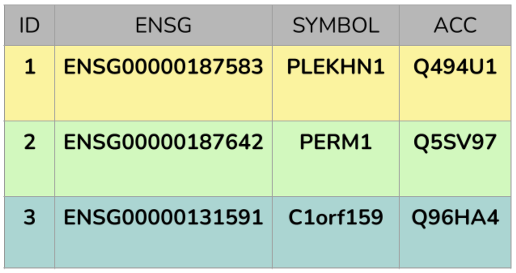
Transformation Rule 1: Projection of Attributes: A triple map may only use a subset of the attributes of a data source, generating thus high overhead whenever the number of attributes used in the triple map and the number of attributes in the data source differs considerably. To illustrate this situation consider the RML triple map in Figure 5.3a whose evaluation produces many duplicates. Additionally, the data source in Figure Figure 5.3c comprises eight attributes but only four attributes are used in the rules. The values of the attributes `ENSG`, `SYMBOL`, `SPECIES`, and `ACC` are repeated, e.g., the rows 1,2, and 3 have the same values in these attributes, and similarly rows 4 and 5, and 6, 7, 8, and 9, respectively. Coincidentally, the evaluation of the triple map in Figure 5.3a creates RDF triples from these four attributes and because during the execution of this triple map the data source is blindly traversed, several duplicated RDF triples are generated. *Transformation rule 1* reduces the overhead caused whenever a triple map utilizes only a subset of the attributes of a data source; it pushes down the projection of the triple map object attributes before the triple map is executed. Thus, during the execution of the triple map only three rows are processed and no duplicated RDF triples are generated. In the case reported in Figure 5.3, processing the original file in Figure Figure 5.3c and the RML triple map (Figure 5.3a) generate five duplicated RDF triples. Contrary, when the file in Figure Figure 5.3b is utilized, no duplicates are produced, thus the overhead during knowledge graph creation is considerably reduced. The time savings are reported in the Chapter 11.

Transformation Rule 2: Pushing Down Projection into Joins: This rule is applied whenever a join exists between two triple maps ma_1 and ma_2 defined over data sources with a large number of attributes that are not utilized in ma_1 and ma_2. To illustrate this case, consider Figure 5.4; the triple maps `TripleMap1` and `TripleMap2` are joined by the join condition highlighted in bold in `TripleMap1`. When this join is executed on datasets in Figures Figure 5.5a and Figure 5.5b, 22 duplicated RDF triples are generated. Duplicate generation considerably impacts the performance of a knowledge graph creation, particularly, whenever duplicates are blindly generated and then, eliminated. To reduce the effect of duplicates during the evaluation of join conditions between two triple maps, MapSDI pushes the projections of the relevant attributes down before the triple maps are executed. As observed in Figure 5.6, this transformation considerably reduces the number of matches of the



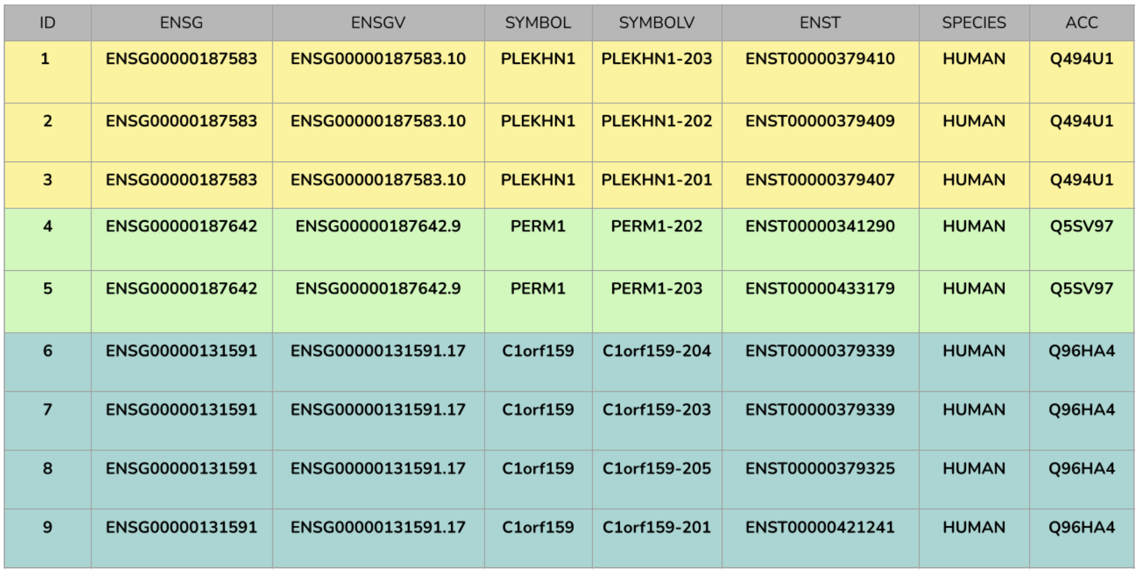
```
<TripleMap1>
  a rr:TriplesMap;
  rml:logicalSource [ rml:source "/data/GenCode-Uniprot.csv";
  rml:referenceFormulation ql:CSV];

    rr:subjectMap [ rr:template "http://project-iasis.eu/Gene/{ENSG}";
    rr:class iasis:Gene];

  rr:predicateObjectMap [ rr:predicate iasis:geneName;
    rr:objectMap [ rml:reference "SYMBOL"];

  rr:predicateObjectMap [ rr:predicate iasis:speciesType;
    rr:objectMap [ rml:reference "SPECIES"];

  rr:predicateObjectMap [ rr:predicate iasis:uniprotID;
    rr:objectMap [ rml:reference "ACC"]; ].
```

(a) Transformation Rule 1



ID	ENSG	SYMBOL	ACC
1	ENSG00000187583	PLEKHN1	Q494U1
2	ENSG00000187642	PERM1	Q5SV97
3	ENSG00000131591	C1orf159	Q96HA4

(b) Portion of a Source File about Genes



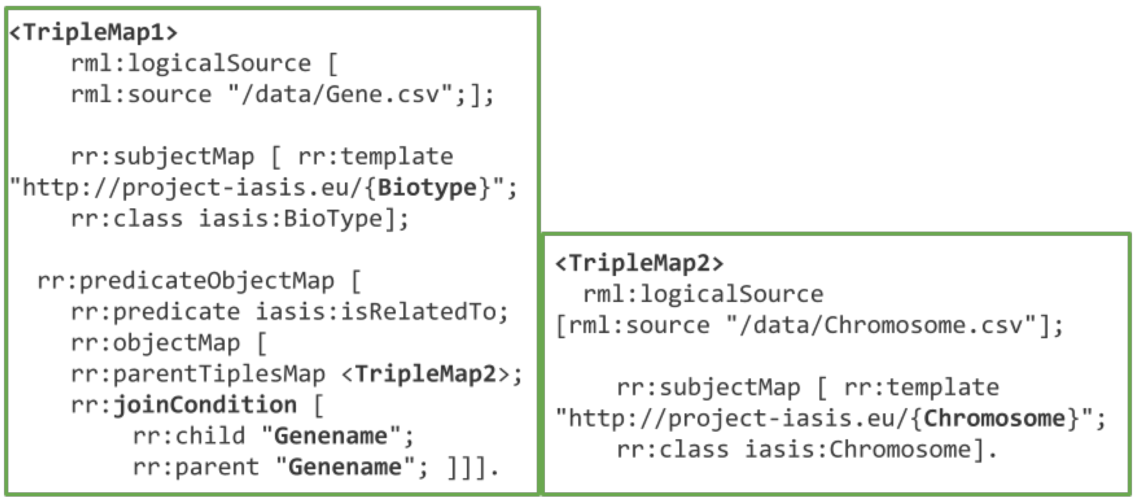
ID	ENSG	ENSGV	SYMBOL	SYMBOLV	ENST	SPECIES	ACC
1	ENSG00000187583	ENSG00000187583.10	PLEKHN1	PLEKHN1-203	ENST00000379410	HUMAN	Q494U1
2	ENSG00000187583	ENSG00000187583.10	PLEKHN1	PLEKHN1-202	ENST00000379409	HUMAN	Q494U1
3	ENSG00000187583	ENSG00000187583.10	PLEKHN1	PLEKHN1-201	ENST00000379407	HUMAN	Q494U1
4	ENSG00000187642	ENSG00000187642.9	PERM1	PERM1-202	ENST00000341290	HUMAN	Q5SV97
5	ENSG00000187642	ENSG00000187642.9	PERM1	PERM1-203	ENST00000433179	HUMAN	Q5SV97
6	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-204	ENST00000379339	HUMAN	Q96HA4
7	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-203	ENST00000379339	HUMAN	Q96HA4
8	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-205	ENST00000379325	HUMAN	Q96HA4
9	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-201	ENST00000421241	HUMAN	Q96HA4

(c) Source File After the Transformation Rule 1

Figure 5.3: **Example of Transformation Rule 1**. Projection of Attributes: (a) RML Triple Map; only four attributes of the file are utilized in the rule; processing the values of these four attributes conduce to the generation of many duplicated RDF triples. (b) A file with information about genes; several values are duplicated across the file. (c) The file resulting from the projection of the attributes utilized in the triple map; the file does not have repeated attributes and the execution of the triple map does not produce duplicated RDF triples.

join condition and the resulting RDF triples.

Once the attributes mentioned in the triple maps in Figure 5.4 are projected out



```
<TripleMap1>
    rml:logicalSource [
    rml:source "/data/Gene.csv";];

    rr:subjectMap [ rr:template
"http://project-iasis.eu/{Biotype}";
    rr:class iasis:BioType];

  rr:predicateObjectMap [
    rr:predicate iasis:isRelatedTo;
    rr:objectMap [
    rr:parentTiplesMap <TripleMap2>;
    rr:joinCondition [
        rr:child "Genename";
        rr:parent "Genename"; ]]].
```

```
<TripleMap2>
  rml:logicalSource
[rml:source "/data/Chromosome.csv"];

    rr:subjectMap [ rr:template
"http://project-iasis.eu/{Chromosome}";
    rr:class iasis:Chromosome].
```

Figure 5.4: **Transformation Rule 2**.

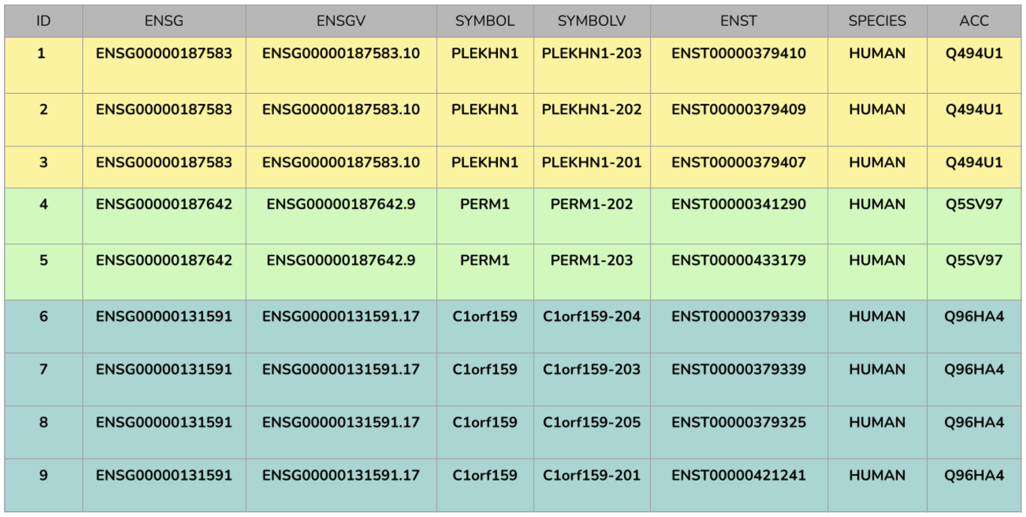
(files in Figures Figure 5.6a and Figure 5.6b), the execution of these triples maps still produces RDF triples that are duplicated. However, the number of duplicates is reduced from 22 to four. Considerably reducing thus, the workload required to generate, check, and eliminate duplicated RDF triples. The results of the experimental study will show the improvements of the MapSDI framework.

Transformation Rule 3: Merging data sources with equivalent attributes: This rule is applied whenever there exist two or more triple mapping rules that generate the same type of subjects associated with the same predicates, but the data is collected from different data sources with attributes that may have different names. This rule allows the MapSDI framework to first, project the relevant attributes, and then merge the data sources; duplicates are eliminated from the merged data source. Additionally, the triple maps are merged in one triple map that will access the merged data source and duplicated RDF triples are not generated (See Figure 5.1).

MapSDI applies the transformation rules 1-3 over the input data integration system $DIS_G = \langle O, S, M \rangle$ in order to generate $DIS'_G = \langle O, S', M' \rangle$; these rules are applied until a fixed point over S' and M' is reached.

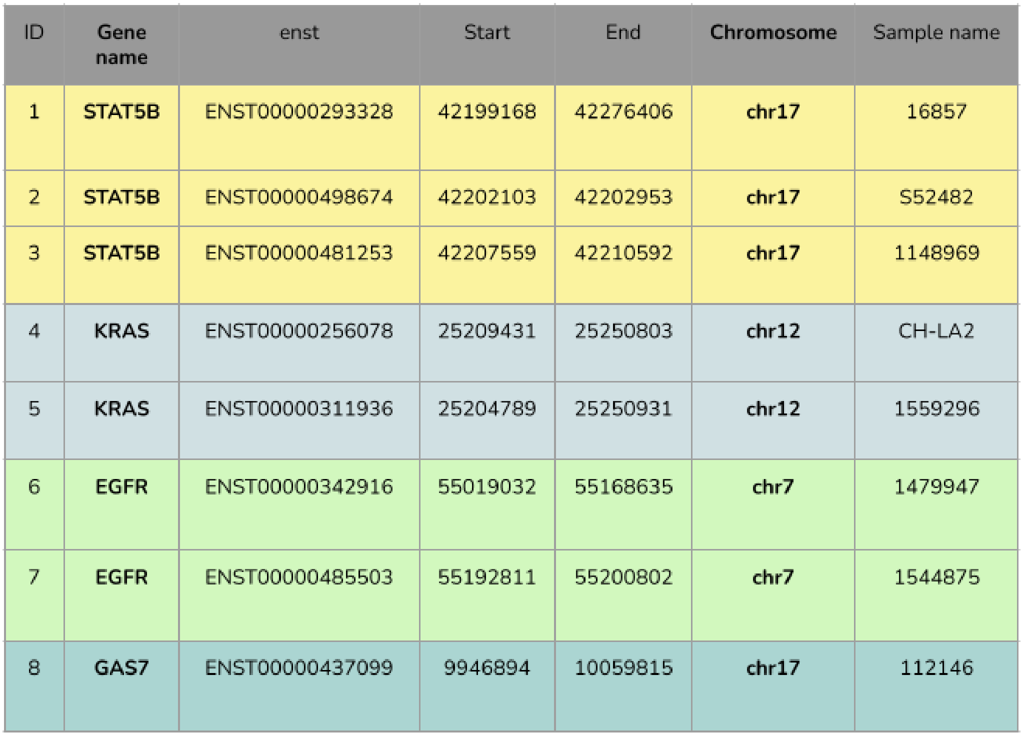
5.2.2 Correctness of Transformation Rules

We demonstrate the correctness of the transformation rules 1-3 by proving that the application of each of these rules preserves the set of RDF triples produced during the evaluation of the original data integration system; these proofs are grounded on



ID	ENSG	ENSGV	SYMBOL	SYMBOLV	ENST	SPECIES	ACC
1	ENSG00000187583	ENSG00000187583.10	PLEKHN1	PLEKHN1-203	ENST00000379410	HUMAN	Q494U1
2	ENSG00000187583	ENSG00000187583.10	PLEKHN1	PLEKHN1-202	ENST00000379409	HUMAN	Q494U1
3	ENSG00000187583	ENSG00000187583.10	PLEKHN1	PLEKHN1-201	ENST00000379407	HUMAN	Q494U1
4	ENSG00000187642	ENSG00000187642.9	PERM1	PERM1-202	ENST00000341290	HUMAN	Q5SV97
5	ENSG00000187642	ENSG00000187642.9	PERM1	PERM1-203	ENST00000433179	HUMAN	Q5SV97
6	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-204	ENST00000379339	HUMAN	Q96HA4
7	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-203	ENST00000379339	HUMAN	Q96HA4
8	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-205	ENST00000379325	HUMAN	Q96HA4
9	ENSG00000131591	ENSG00000131591.17	C1orf159	C1orf159-201	ENST00000421241	HUMAN	Q96HA4

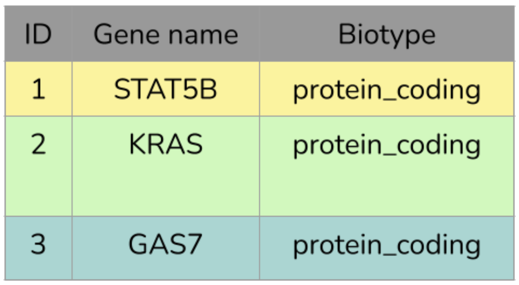
(a) Portion of a Source File about Genes (Outer Source File)



ID	Gene name	enst	Start	End	Chromosome	Sample name
1	STAT5B	ENST00000293328	42199168	42276406	chr17	16857
2	STAT5B	ENST00000498674	42202103	42202953	chr17	S52482
3	STAT5B	ENST00000481253	42207559	42210592	chr17	1148969
4	KRAS	ENST00000256078	25209431	25250803	chr12	CH-LA2
5	KRAS	ENST00000311936	25204789	25250931	chr12	1559296
6	EGFR	ENST00000342916	55019032	55168635	chr7	1479947
7	EGFR	ENST00000485503	55192811	55200802	chr7	1544875
8	GAS7	ENST00000437099	9946894	10059815	chr17	112146

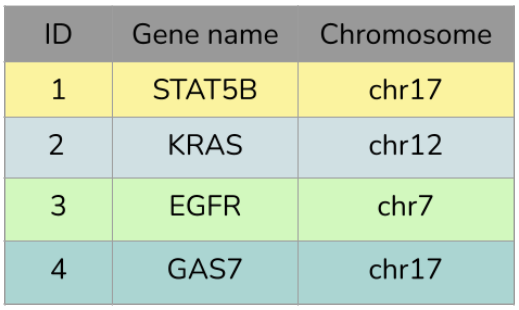
(b) Portion of the Source File about Chromosomes (Inner Source File)

Figure 5.5: **Example of Transformation Rule 2**. Pushing down Projections into a Join: (a) and (b) Files containing data to be considered as the outer and inner data sources of `TripleMap1` (Figure 5.4), respectively. Duplicates in the join attribute conduce the generation of 22 duplicated RDF triples.



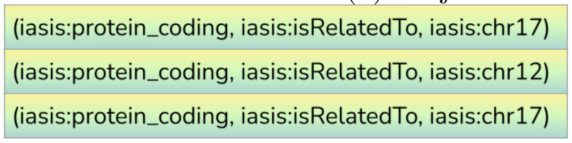
ID	Gene name	Biotype
1	STAT5B	protein_coding
2	KRAS	protein_coding
3	GAS7	protein_coding

(a) Projection on Genes



ID	Gene name	Chromosome
1	STAT5B	chr17
2	KRAS	chr12
3	EGFR	chr7
4	GAS7	chr17

(b) Projection on Chromosomes



(iasis:protein_coding, iasis:isRelatedTo, iasis:chr17)
(iasis:protein_coding, iasis:isRelatedTo, iasis:chr12)
(iasis:protein_coding, iasis:isRelatedTo, iasis:chr17)

(c) RDF triples with reduced duplicates

Figure 5.6: **Example of Transformation Rule 2**. Pushing down Projections into a Join: (a) and (b) Projecting out from files in Figures Figure 5.5a and Figure 5.5a the attributes mentioned in triple maps in Figure Figure 5.4. (c) RDF triples are produced by the triple maps over the projected attributes; duplicates are reduced from 22 to four.

the axiomatic system of the Relational Algebra [62].

Transformation Rule 1: Projection of Attributes.

For each mapping assertion ma_i in M with sources $S_z(\overline{X_z})$ in the body of ma_i, the transformation rule 1, adds new sources S'_z to S', in the way, that S'_z is equal to $\prod_{Att} S_z$ and Att is the set of attributes utilized in $\overline{X_z}$. The mapping assertion ma_i is removed from M' and a new mapping assertion ma'_i where all the sources $S_z(\overline{X_z})$ are replaced by $S'_z(\overline{X_z})$ is added to M'. Since the attributes from the sources S_z used in $\overline{X_z}$ are maintained in the new data sources S'_z and in the mapping assertion ma'_i, the results of $\mathit{RDFize}(DIS_G = \langle O, S, M\rangle)$ and $\mathit{RDFize}(DIS'_G = \langle O, S', M'\rangle)$ are the same.

Transformation Rule 2: Pushing Down Projection into Joins. Transformation rule 2 is applied over a mapping assertion ma_i whenever there exist attributes and variables in the sources of the body of ma_i that are neither applied by the *terms* in the head of ma_i nor to join the data sources in the body of ma_i, i.e., $\theta(\overline{X}_{i,1}, \overline{X}_{i,2})$. If so, transformation rule 2 projects out from the sources $S_z(\overline{X_z})$ in the body of ma_i, all the attributes and variables that are required. Formally, the rewriting of ma_i is defined as follows: Let $\overline{Z}$ be the set of variables in the head of ma_i or in the join of at least two sources in the body i.e., $\theta(\overline{X}_{i,1}, \overline{X}_{i,2})$.

$$S_i(\overline{X_{i,1}}), S^M A_j(\overline{X_{i,2}}), \theta(\overline{X}_{i,1}, \overline{X}_{i,2}) : -P(f_1(t_1), f_2(t_2))$$

The application of the transformation rule 2, replaces ma_i by the rule ma'_i:

$$S_i(\overline{X'_{i,1}}), S_j^{MA}(\overline{X'_{i,2}}), \theta(\overline{X'}_{i,1}, \overline{X'}_{i,2}) : -P(f_1(t_1), f_2(t_2))$$

Where each $X'_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq 2$, is defined as follows:

$$X'_{i,j} = X_{i,j} - \{(att_{i,j}, X_{i,j}) \mid (att_{i,j}, X_{i,j}) \in X_{i,j} \; and \; X_{i,j} \notin \overline{Z}\}$$

The transformation 2 is grounded on the axiomatic system of the Relational Algebra, specifically, on the rule axiom that states the properties of distributing the Project operator over a Join (rule number 8 in [62]). Thus, after applying this transformation rule and replacing ma_i by ma'_i in M', the results of $\mathit{RDFize}(DIS_G = \langle O, S, M\rangle)$ and $\mathit{RDFize}(DIS'_G = \langle O, S', M'\rangle)$ are the same.

Transformation Rule 3: Merging data sources with equivalent attributes. This rule is applied over two mapping assertions, ma_i and ma_j, whenever both mapping assertions share the same head but the bodies are composed of different data sources, i.e., $S_i(\overline{X_i}) : -head$ and $S_j(\overline{X_j}) : -head$. The result of applying the transformation rule 3 is a new data source $S_{i,j}$ that is populated with values of the attributes from S_i and S_j that are required for the *head*. Moreover, ma_i and ma_j are replaced by the mapping assertion $ma_{i,j}$ in M', $S_{i,j}(\overline{X}_{i,j}) : -head$

- $S_{i,j}$ is the union of $\prod_{Att_i} S_i$ and $\prod_{Att_j} S_j$ such that Att_i and Att_j, respectively, are the attributes in $\overline{X_i}$ and $\overline{X_j}$ related with variables in the *head*.

- The projected attributes in $S_{i,j}$ are renamed and these new attributes are used in $\overline{X_{i,j}}$ associated with the corresponding variables in the *head*.

The transformation 3 is also supported on the axiomatic system of the Relational Algebra, specifically, on the rule axiom that states the properties of distributing the Project operator over a Union (rule number 12 in [62]). Thus, after applying this transformation rule and replacing ma_i and ma_j by $ma_{i,j}$ in M', and adding the data source $S_{i,j}$ to S', the results of $RDFize(DIS_G = \langle O, S, M\rangle)$ and $RDFize(DIS'_G = \langle O, S', M'\rangle)$ are the same.

5.3 Summary

In this chapter, we address the problem of optimizing semantically integrating data into a knowledge graph. We present MapSDI, a framework devised for enabling the semantic enrichment of data characterized by the dominant dimensions of big data, i.e., volume, variety, and veracity. MapSDI resorts to the properties of the relational algebra operators and to the knowledge encoded in the mapping rules to identify the transformations that need to be performed to the input data to empower the performance of existing knowledge graph creation frameworks. Thus, our resource broadens the repertoire of techniques available to integrate heterogeneous datasets into a knowledge graph, and we hope that these techniques help the community in the development of more scalable knowledge graph-based applications.

Chapter 6

Planner: Sourse-based Optimizations

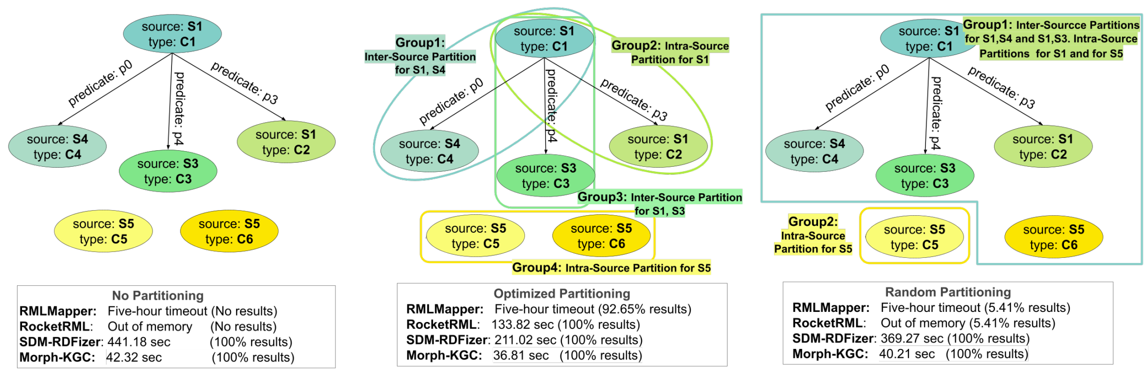
Similar to the previous chapter, this chapter also tackles the problem of efficient data integration materialization while transforming heterogeneous data into RDF triples. Accordingly, this chapter provide another solution to the following research question:

RQ3: How can efficiency be ensured in a materialized data integration system?

We target the mentioned research question from inter-mapping assertions perspective. In other words, we propose the techniques for planning a given set of mapping assertions to provide an optimized execution plan by partitioning and scheduling the execution of the assertions. The contents of this chapter are the results of my collaboration with my supervisor and another colleague. The contents of this paper are published in [37].

6.1 Motivating Example

We motivate our work, illustrating the challenges that the execution of mapping assertions brings to the process of KG creation from multiple data sources. Continuous creation and maintenance of KGs demand scalability in terms of required execution time. Figure 6.1 presents three configurations of a set of mapping assertions that define a KG $G1$. The set comprises mapping assertions specifying the properties and attributes of five classes (`C1`,`C2`,`C3`,`C4`, and `C5`) over four data sources (`S1`, `S3`, `S4`, and `S5`). These data sources correspond to the SDM-Genomic-Datasets, each containing


source: S1 type: C1
predicate: p0
predicate: p4
predicate: p3
source: S4 type: C4
source: S3 type: C3
source: S1 type: C2
source: S5 type: C5
source: S5 type: C6
Group1: Inter-Source Partition for S1, S4
Group2: Intra-Source Partition for S1
Group3: Inter-Source Partition for S1, S3
Group4: Intra-Source Partition for S5
Group1: Inter-Sourcce Partitions for S1,S4 and S1,S3. Intra-Source Partitions for S1 and for S5
Group2: Intra-Source Partition for S5

No Partitioning
RMLMapper: Five-hour timeout (No results)
RocketRML: Out of memory (No results)
SDM-RDFizer: 441.18 sec (100% results)
Morph-KGC: 42.32 sec (100% results)

Optimized Partitioning
RMLMapper: Five-hour timeout (92.65% results)
RocketRML: 133.82 sec (100% results)
SDM-RDFizer: 211.02 sec (100% results)
Morph-KGC: 36.81 sec (100% results)

Random Partitioning
RMLMapper: Five-hour timeout (5.41% results)
RocketRML: Out of memory (5.41% results)
SDM-RDFizer: 369.27 sec (100% results)
Morph-KGC: 40.21 sec (100% results)

Figure 6.1: **Motivating Example**.

one Million records and up to 15 attributes.

The configuration `No Partitioning` depicts all the mapping assertions; they are executed together on four state-of-the-art [R2]RML-compliant engines, RMLMapper [22], RocketRML [63], SDM-RDFizer [36], and Morph-KGC [3]. Executing all the assertions together demands from each engine, data management techniques like the ones implemented by Morph-KGC. These techniques must allow planning both the execution of the mapping assertions and the period to maintain in memory each source. Unfortunately, RMLMapper and RocketRML are not as scalable as Morph-KGC and cannot produce any results. RocketRML ran out of memory, while RMLMapper timed out after five hours. On the contrary, all the engines exhibit better performance when the assertions are divided into intra- and inter-source partitions and executed in plans generated based on these partitions; the improvement, albeit not so significant as in the other engines, can also be observed in Morph-KGC. First, when four groups of partitions are created (i.e., `Optimized Partition`), the performance of the four engines is empowered, and three of them can generate 100% of the results. Each group comprises one intra-source partition of a source S_j and at most one inter-source partition of another source S_i to S_j. Moreover, the groups are executed in parallel. Lastly, the execution of the configuration named, `Random Partition`, indicates that no combination of the intra- and inter-source partitions leads to efficient mapping assertions plans. In this case, `Group1` includes two inter- and four intra-source partitions, while `Group2` comprises only one intra-source partition. Although `Group2` is executed by all the engines, RMLMapper and RocketRML could not produce any result during the execution of `Group1`, and they could only produce 5.41% of the total number of RDF triples. This paper addresses the challenges of generating plans of mapping assertions that empower [R2]RML engines and

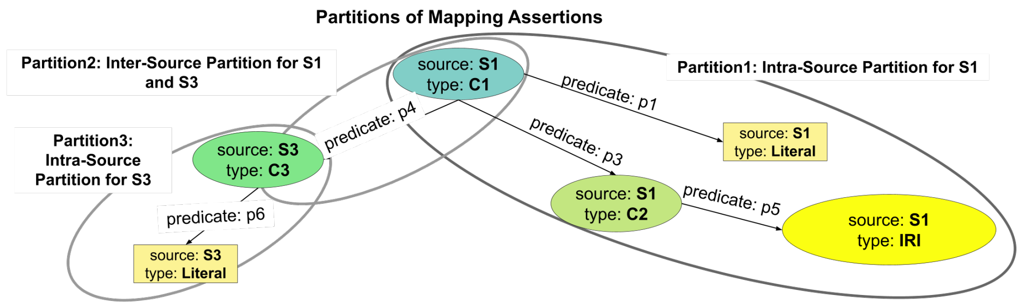


Figure 6.2: **Partitioning of Mapping Assertions**.

enhance their scalability during KG creation.

6.2 The *Planner*

planner assesses an optimized number of partitions considering the number of data sources, type of mapping assertions, and the associations between different assertions. After providing a list of partitions and assertions that belong to each partition, the *planner* determines their execution order.

As observed in Figure 6.1, the order and grouping of the mapping assertions impact the execution time of the engines, which is crucial to enable the generation of results in real-world scenarios.

6.2.1 Partition of Mapping Assertions

In a data integration system $DIS_{\mathcal{G}} = \langle O, S, M \rangle$, the mapping assertions in M can be grouped to create a partition of M. We define two types of partitions: **Intra-source** and **Inter-source** mapping assertion partitions. Given a source S_k in S, an **Intra-source** partition for S_k corresponds to a set of all the mapping assertions that have only the source S_k in the body clause, i.e., it comprises concept, attribute, single-source role, and referenced-source role mapping assertions over S_k. An **Inter-source** groups mapping assertions of two sources S_i and S_j which are related via multi-source role mapping assertions. Figure 6.2 presents three partitions for mapping assertions in the running example. To increase readability, mapping assertions are depicted in a directed graph where directed edges represent predicates defined by mapping assertions (i.e., `p4`, `p6`, `p1`, `p3`, and `p5`). A node denotes a logical source

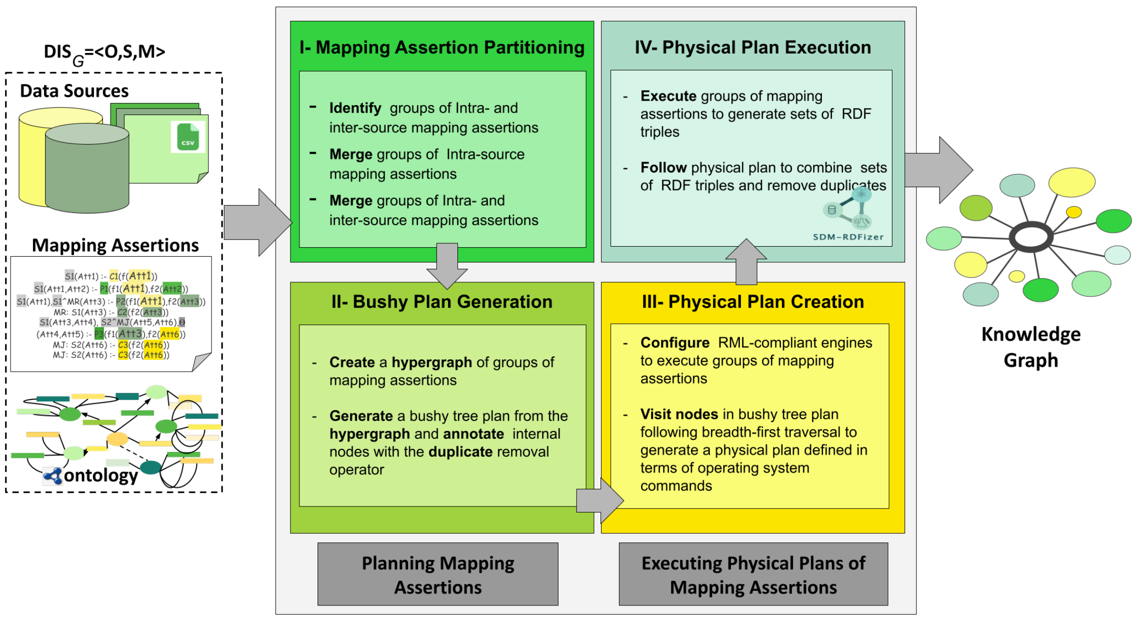

DIS_G=<O,S,M>
Data Sources
Mapping Assertions
ontology
I- Mapping Assertion Partitioning
Identify groups of Intra- and inter-source mapping assertions
Merge groups of Intra-source mapping assertions
Merge groups of Intra- and inter-source mapping assertions
II- Bushy Plan Generation
Create a hypergraph of groups of mapping assertions
Generate a bushy tree plan from the hypergraph and annotate internal nodes with the duplicate removal operator
III- Physical Plan Creation
Configure RML-compliant engines to execute groups of mapping assertions
Visit nodes in bushy tree plan following breadth-first traversal to generate a physical plan defined in terms of operating system commands
IV- Physical Plan Execution
Execute groups of mapping assertions to generate sets of RDF triples
Follow physical plan to combine sets of RDF triples and remove duplicates
SDM-RDFizer
Planning Mapping Assertions
Executing Physical Plans of Mapping Assertions
Knowledge Graph

Figure 6.3: **Architecture of the *Planner*.**

and the type of the mapped entity. All the assertions defined over $S1$ (resp. $S3$) are grouped together into `Partition1` (resp. `Partition3`). Moreover, there is only one assertion between `S1` and `S3`, thus, `Partition2` is an **inter-source** partition and comprises the multi-source mapping assertion for `p4` and the concept mapping assertion that defines the class `C3`.

6.2.2 Problem Statement

The aim is to generate GP_M, a set of sets of mapping assertions in M (inter- and intra-source), such as the union of all the sets in GP_M is equal to M, and the pairwise intersection of the sets in GP_M is empty. That is, GP_M is a partition of M. Moreover, since the order in which the groups in GP_M may also impact, we define a plan $\overline{GP}_M$ over the groups in GP_M, as a bushy tree plan of the groups in GP_M, where each internal node represents the union operator that merges the RDF triples produced during the execution of each group in GP_M. Lastly, since results produced during the execution of the GP_M groups may overlap, duplicate removal may be required at different steps of the execution of $\overline{GP}_M$. Thus, each node is annotated with the union operator, which merges the inputs and produces the results.

6.2.3 Proposed Solution: the *Planner*

We propose a heuristic-based approach to generate a bushy tree $\overline{GP}_M$ that corresponds to a solution to the problem of *planning KG creation*. The execution of intra- and inter-source groups of mapping assertions independently induces source-based scheduling of the execution of the mapping assertions. Moreover, the duplicate removal operators are pushed down into the bushy tree following an eager execution of duplicate removal. As a result, the union operators are scheduled over small sets of RDF triples, and the effect of merging multisets of RDF triples is mitigated. Then, $\overline{GP}_M$ is translated into a physical plan defined in terms of operating system commands. It schedules the execution of each group of mapping assertions and union operators according to $\overline{GP}_M$. In the following, we explain the pipeline of the *planner*. The pipeline comprises, first, the phase of planning where the bushy tree is created, and then, the execution phase, where $\overline{GP}_M$ is translated into a physical plan and executed over a particular RML-compliant engine.

Planning Mapping Assertions

This step comprises the components of mapping assertion partitioning and bushy plan generation. The algorithm receives a data integration system $DIS_{\mathcal{G}} = \langle O, S, M \rangle$ and partitions M into groups of intra- and inter-source mapping assertions. Then, they are heuristically combined into a bushy tree plan.

Mapping Assertion Partitioning. This component describes the algorithm that receives as input the set of mapping assertions M and initializes GP_M with the intra- and inter-source mapping assertion partitions of M. Then, it *greedily* decides to combine two groups g_i and g_j in GP_M into a group $g_{i,j}$ whenever any of the following conditions are satisfied:

- *Merging Intra-Source Partitions.* Suppose g_i and g_j only comprise intra-source mapping assertion partitions of sources S' (i.e., $S' \subseteq S$). Additionally, there are no sources S_i and S_j in S' such that there exists in GP_M an inter-source assertion mapping partition for S_i and S_j. Then, groups g_i and g_j can be merged into the group $g_{i,j}$ in GP_M; $g_{i,j}$ comprises intra-source assertion mapping partitions in g_i and g_j.
- *Merging Inter- and Intra-Source Partitions.* Suppose the group g_i comprises an inter-source mapping partition for S_i and S_j, where S_j is the referenced source (i.e., logical source of the parent triples map). Additionally, the group g_j only includes the intra-source mapping assertion of S_j. Thus, g_i and g_j

can be merged into the group $g_{i,j}$ in GP_M. The group $g_{i,j}$ only includes intra-source assertion mapping partitions of S_j and the inter-source partition for S_i and S_j. In case S_j is the referenced source of various inter-source mapping partitions, the intra-source mapping assertion partition of S_j is only combined with one inter-source partition. The selection is done randomly. The selected combination of the intra- and inter-source mapping partitions may be more expensive than other options. As a result, this decision may negatively impact the performance of a bushy tree plan.

The algorithm iterates until a fixed-point is reached over GP_M, i.e., an iteration of the algorithm where all the pairs of groups g_i and g_j are revised, and no new group $g_{i,j}$ can replace them in GP_M. Finally, a bushy tree $\overline{GP}_M$ for the groups GP_M of mapping assertion partitions is generated following a greedy heuristic-based algorithm and assumes that sub-plans produced so far, are optimal. Also, the algorithm combines the first groups of partitions whose union requires duplicate removal.

Executing Mapping Assertions

This step receives a bushy tree $\overline{GP}_M$, and generates a physical plan that can execute the mapping assertions in M following the order stated in $\overline{GP}_M$. Figure 6.3 depicts the main two components of this step of the pipeline including **physical plan creation** and **physical plan execution**. First, nodes in $\overline{GP}_M$ are visited following a breadth-first traversal to generate a physical plan. A physical plan is defined in terms of operating system commands that enable the execution of a [R2]RML-compliant engine calls to evaluate a group of mapping assertions and generate RDF triples that will be part of a KG.

6.3 Summary

In this chapter, we tackle the problem of efficient KG creation. We present heuristic-based solutions that can identify execution plans that can efficiently generate KGs. The execution planning techniques partition mapping assertions and schedule them into execution plans that reduce execution time. Thus, the proposed planning methods evidence the crucial role that optimization techniques, defined in the context of query processing, also have in the KG creation process.

Chapter 7

EABlock and *GenoConductor*: Declarative Data Operation Function Libraries

Despite encoding the enormous amount of rich and valuable data, existing data sources are mostly created independently, being a significant challenge to their integration specifically at the data-level. For instance in the biomedical domain, there exist many publicly available data and knowledge bases providing the results of decades of research on clinical and biological aspects of genetic variants and their association with different diseases. Accordingly, gaining broad insight into the role of different variant in diagnosis, prognosis, and treatment of diseases is only achievable by analyzing existing data and knowledge bases integrated; rather than linked.

Mapping languages, such as RML and R2RML, facilitate declarative specification of the process of applying meta-data and integrating data into a knowledge graph. Mapping assertions can also include knowledge extraction and data operation functions in addition to expressing correspondences among data sources and a unified schema. Including user-defined functions in mapping assertions represents a powerful formalism to specify pipelines for integrating data into a knowledge graph transparently. Surprisingly, these formalisms are not fully adapted, and many knowledge graphs rely on executing *ad-hoc* programs as pre/post-processing to integrate data.

In this chapter we contribute to the following research question:

RQ2: How to merge data operators and sources in a materialized data integration system?

The two main contributions of this chapter and the challenges we tackle are explained in the following. **I.** we introduce *EABlock*, a library of user-defined functions to perform Entity Alignment (EA) as part of RML mapping assertions; performing entity recognition from textual attributes and linking the recognized entities to the corresponding resources in Wikidata, DBpedia, and domain specific thesaurus, e.g., UMLS. **II.** we tackle a domain-specific challenge, the problem of reconciling heterogeneous representations of the genomic variants across different sources. There are several parallel efforts from different organizations to manually curate the available evidence of associations between genomic variations and different cancers. However, the only option to detect the overlaps between different provided data/knowledge bases is to harmonize and integrate variations in order to be able to study all the associations as a whole. Hence, we introduce *GenboConductor*, a user-defined functions library specialized to align genomic variations across different sources. The functions of *EABlock* and *GenboConductor* are publicly available through Dragoman GitHub repository as they can be executed by *Dragoman* efficiently. Moreover, the content of this chapter is partially published in [42].

7.1 Motivating Example

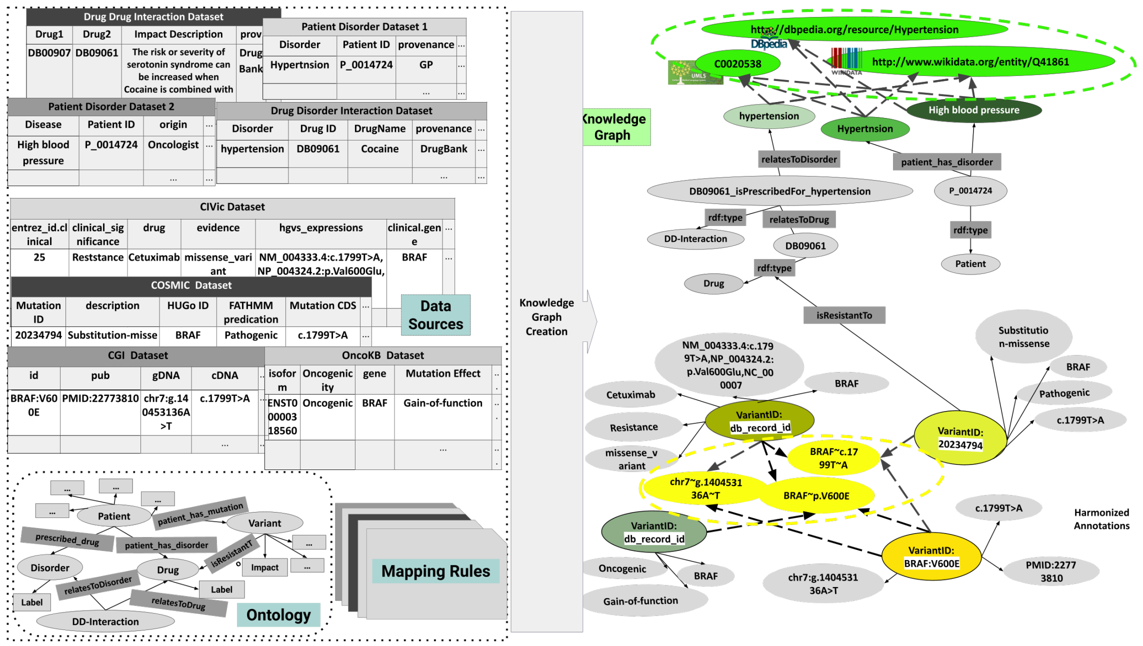
We illustrate the motivation behind developing *EABlock* and GenoConductor with a mock example from a real-world scenario in Figure 7.1. In this scenario, the aim is to integrate eight datasets obtained from different sources into a knowledge graph. The datasets consist of **a)** Patient data extracted from two different clinical notes provided by a general practitioner (GP) and an oncologist including the comorbidities from which the patient is suffering, **b)** The drug related data extracted from DrugBank[9] including drug-drug-interaction providing information on the possible interactions between different drugs and the impact on effectiveness of each, and drug-disorder data revealing information on list of drugs that can be prescribed for each disorder. **c)** Four datasets related to the genomic variation data received from different publicly available data/knowledge bases including COSMIC[10], CIViC[11], CGI[12], and

[9] `https://go.drugbank.com/`

[10] `https://cancer.sanger.ac.uk/cosmicGRCh37,version90,releasedAugust2019`

[11] `https://civicdb.org/`

[12] `https://www.cgi.com/en`


Drug Drug Interaction Dataset

Drug1	Drug2	Impact Description	prov
DB00907	DB09061	The risk or severity of serotonin syndrome can be increased when Cocaine is combined with	Drug Bank

Patient Disorder Dataset 1

Disorder	Patient ID	provenance	...
Hypertnsion	P_0014724	GP	...
		...	...

Patient Disorder Dataset 2

Disease	Patient ID	origin	...
High blood pressure	P_0014724	Oncologist	...
		...	...

Drug Disorder Interaction Dataset

Disorder	Drug ID	DrugName	provenance	...
hypertension	DB09061	Cocaine	DrugBank	...
			...	...

CIVic Dataset

entrez_id.clinical	clinical_significance	drug	evidence	hgvs_expressions	clinical.gene	...
25	Reststance	Cetuximab	missense_variant	NM_004333.4:c.1799T>A, NP_004324.2:p.Val600Glu,	BRAF	...

COSMIC Dataset

Mutation ID	description	HUGo ID	FATHMM predication	Mutation CDS	...
20234794	Substitution-misse	BRAF	Pathogenic	c.1799T>A	...

CGI Dataset

id	pub	gDNA	cDNA	...
BRAF:V600E	PMID:22773810	chr7:g.140453136A>T	c.1799T>A	...
			...	...

OncoKB Dataset

isoform	Oncogenicity	gene	Mutation Effect	...
ENST000003 18560	Oncogenic	BRAF	Gain-of-function	...
			...	...

Figure 7.1: **Motivating Examples of Entity Alignment Function Libraries**. Data integration from eight datasets. Starting from the above, different interoperability issues observed in the first four datasets: concept disorder is modeled differently in the dataset one and two. The entity hypertension is represented with various entities, and its name is misspelled in the dataset one. The main and important interoperability issue in the other four datasets that correspond to the genomic data is the lack of a unified identifier. In both cases, entity alignment is required to be performed enabling conflict resolution and integration into a knowledge graph. problems. In the first case, entity alignment is performed with UMLS, DBpedia, or Wikidata, while, in the case of variants resolution, synthesized identifiers extracted from available data are applied.

OncoKB[13]. A portion of the KG created by a naive approach can be observed in Figure 7.1. A closer look reveals that the same disorder instance exists as three separated nodes in the graph (shown with different green colors), i.e., there is an interoperability conflict among them. The existing interoperability issue can be traced back to the raw data where **I.** the same disorder is represented with different names by clinical physiologists, and **II.** the name of the disorder is misspelled in one of the records. An important point is regarding the connection between the instances of

[13] https://www.oncokb.org/

the generated KG and instances in available domain-specific sources (e.g., UMLS) or encyclopedic KGs (e.g., DBpedia and Wikidata) which represent the same real-world entities. More specifically, annotating the instances of the generated knowledge graph with the instances in UMLS, DBpedia, and Wikidata is a solution to align the differently represented instances in a knowledge graph.

Another unintegrated portion of the generated knowledge graph corresponds to the instances of the class Variation. As it can be shown in the Figure 7.1 in different shades of yellow, there appear four various instances of the same variant related to the gene "BRAF" in the knowledge graph. The reason can be explained based on the fact that variants are represented in different formats across different sources. There has been different attempts in providing guidelines [26] and standards for variant nomenclature and descriptions [27]. However with the rapid generation of data, there are always large groups of conceptualization of variants across different consortiums, tools, and data/knowledge bases in which standard formats are not feasible [79]. Additionally, albeit following the standard guidelines, a comprehensive study of the genetic variants requires a joint analysis of different "omics" layers (genomics, transcriptomics, proteomics, etc). There are ontologies such as SNP Ontology[14] confirming this important fact about variations by creating one specific class i.e., "reference_variant" to only represent the variation in genomic level on the complete chromosome sequence. These confound the identification of variants across different sources. Moreover, the lack of a unified identifier causes entity matching methods and several attempts of data integration systems to fail. For instance the approach proposed by Jha et al. [40] is only able to interlink variants represented with the same identifier. Nevertheless, as it can be observed in Figure 7.1, considering three unified synthesized representations i.e., each for one of the omics layer descriptions, and annotating the instances with at least one of them, the instances can be aligned. Both observations emphasize the importance of including entity alignment as a module in the pipeline of KG creation. It should be noted that following FAIR principles [76], transparency and reproducibility are essential requirements in pipelines of KG creation. All blocks applied as part of the main process or pre- or post-processing of KG creation should be transparent and traceable. By virtue of having the possibility of integrating entity alignment functions in mapping rules, we focus on developing function libraries compliant with RML+FnO.

[14]`https://bioportal.bioontology.org/ontologies/SNPO`

7.2 ***EABlock***: An Entity Alignment Function Library Applying UMLS, DBpedia, and Wikidata

Problem Statement: As shown in Figure 7.1, a knowledge graph can comprise entities that correspond to the same real-world entity (e.g., various entities representing hypertension). We address the problem of aligning entities in a KG G_1=(O_1,V_1,E_1) with entities in an existing KG G_2=(O_2,V_2,E_2) efficiently. Encyclopedic KGs like DBpedia [4] or Wikidata [74], or domain-specific (e.g., UMLS [7]) correspond to KGs G_2 against where the alignment is performed. **Proposed Solution:** Entity alignment from G_1 to G_2, $\gamma(G_1 \mid G_2)$, is defined in terms of an *ideal KG*, $G^* = (O^*, V^*, E^*)$, that includes the nodes and edges in G_1 and G_2 plus all the edges that relate nodes in G_1 with nodes in G_2. A solution to $\gamma(G_1 \mid G_2)$ corresponds to a *maximal partial function* ζ:$V_1 \to V_2$ such that $\gamma(G_1 \mid G_2, \zeta)$=$\{(s_1, sameAs, \zeta(s1)) \mid (s_1, sameAs, \zeta(s1)) \in E^*\}$[15]. $DE_{G_{1,2}} = \langle DataSets_1, DataOperators, Meta\text{-}Data_1, Mappings_{1,2}\rangle$ defines the KG, $G_{1,2}$=$(O_1 \cup \{sameAs\}, V_1 \cup V_2, E_1 \cup \gamma(G_1 \mid G_2, \zeta))$. The set $Mappings_{1,2}$ is a superset of $Mappings_1$ including all triples maps that define ζ and enable the computation of $\gamma(G_1 \mid G_2, \zeta)$.

EABlock proposes a computational block to solve entity alignment over textual attributes providing a set of data operation functions relying on an entity and relation linking tool. **a)** *EABlock* links entities encoded in labels and short text to controlled vocabularies described by meta-data and resources in encyclopedic and other domain-specific knowledge graphs. For this purpose, *EABlock* introduces a set of operating functions resorting to an entity and relation linking tool. **b)** *EABlock* functions are defined in a human and machine-readable medium, meeting the requirements of meta-data in terms of transparency and reusability. Although the outcome of *EABlock* representing the aligned entities and annotations provides meta-data for the knowledge graph, the addition of *EABlock* functions to the meta-data of the DE equips this layer for further reproduction or maintenance of the knowledge graph with newly added data. **c)** *EABlock* functions can be easily integrated into the mappings expressing the relations among the data and the ontology using RML language, applying available extensions of the language. **d)** *EABlock* also provides an efficient evaluation strategy to materialize the calls of the functions in the mappings extending data sources and transforming mappings to function-free RML mappings that are adaptable by any RML-compliant knowledge graph creation pipeline.

[15]A partial function ζ:$V_1 \to V_2$ is a function from a subset of the V_1. ζ is maximal in the partial ordered set of all the functions from $V_1 \to V_2$.

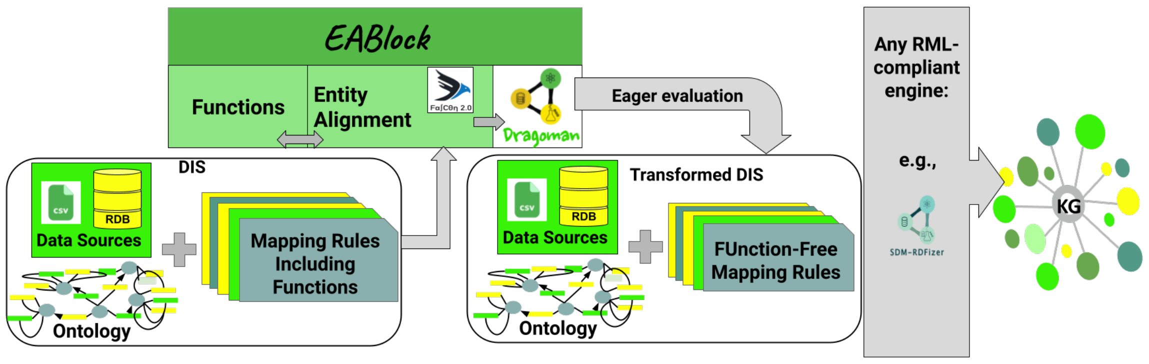


Figure 7.2: **The architecture and the application of EABlock**.EABlock is comprised of a set of FnO functions that resorts to an Entity Alignment engine. Relying on Dragoman, an RML+FnO interpreter, the *EABlock* functions included in RML+FnO mapping rules are executed and transformed into function-free rules. Then the transformed data integration system can be translated into an RDF knowledge graph utilizing any RML-compliant engine such as SDM-RDFizer.

As shown in Figure 7.2, *EABlock* composes a set of functions signatures in FnO. The functions can be divided into two categories based on their domains and ranges. **Keyword-based** functions receive case-insensitive keywords as input and generate one entity as the output, and **Short text-based** functions accept a case-insensitive short text as input and output a list of entities. To perform the task of entity alignment including the Named Entity Recognition (NER) and Entity Linking (EL) tasks, EABlock functions rely on Falcon2.0 [57] through an API. The EABlock functions resorts Dragoman to execute the functions and retrieve the results of the entity alignment generated by the entity alignment tool following an eager evaluation strategy. The eager evaluation strategy gives the basis for an efficient and RML engine-agnostic execution of the *EABlock* functions. Depending on the category of the function, the output of the *EABlock* functions is one of the followings. **a.** If the function is a **Keyword-based** function, for each input value, one record is added to the output dataset. **b.** However, if the function is **Short text-based**, after evaluation of the function and receiving the list of linked entities, *EABlock* generates the output dataset including one record for each entity in the list of linked entities, i.e., for each entity in the list of the retrieved linked entities, one record is added to the output dataset which includes input value and the linked entity. In this way, it is ensured that the generated datasets can be translated by any RML-compliant engine and result in exactly the same RDF triples; since different RML engines may have different interpretations of an RDF list.

Implementation and Application *EABlock* approach is implemented in Python 3.6, publicly available via [16], and compatible with being executed by Dragoman. As a proof of concept, *EABlock* integrates Falcon2.0 API [17] to perform the NER and EL tasks.

7.3 *GenoConductor*: A Function Library to Align Genomic Variations

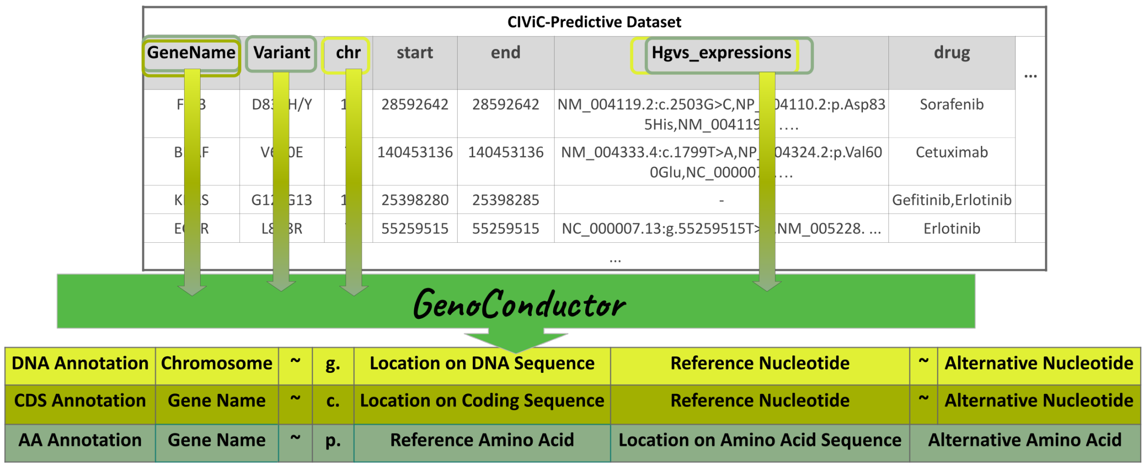
As we observed in the example Figure 7.1, genomic variants can be presented in various formats in different databases. We introduce three harmonized representations of genomic variants to provide identical annotations of the same variants at different sequencing levels according to the data mutually provided by different sources. These harmonized representations including "DNA Annotation", "CDS Annotation", and "AA Annotation" are defined to be compatible with Human Genome Variation Society guideline[26] for describing variants and the Uniform Resource Identifier (URI) syntax standard provided by Berners-Lee et al. [18]. Each harmonized representation provides information regarding one of the following concepts. **a.** Nucleotide variation on DNA (both coding and non-coding) comprises chromosome number, alteration position on DNA, reference nucleotide, and altered nucleotide. **b.** Nucleotide variation on coding DNA including gene name, alteration position on coding DNA, reference nucleotide, and altered nucleotide. **c.** Amino acid variation on peptide sequence consists of gene name, reference amino acid, alteration position on peptide sequence, and altered amino acid. To generate mentioned harmonized representations, depending on the schema of each dataset, a set of data operation functions is required. The functions of *GenoConductor* are designed to extract specific data values required to assemble the harmonized representations from data sources. Figure 7.3 illustrates with an example the data that is extracted and assembled from the attributes of a data source to build the harmonized representations.

GenoConductor is a library of data operation functions defined with FnO signatures. Functions of *GenoConductor* can be applied in the RML mapping rules as part of the curation and semantification process in order to facilitate the **traceability** and **reproducibility**. An important feature of *GenoConductor* is that the provided

[16]`https://github.com/SDM-TIB/Dragoman/blob/master/Sources/FunctionLibraries/functions_GenoConductor.py`

[17]`https://labs.tib.eu/sdm/falconmedical/falcon2/`

[18]`https://tools.ietf.org/html/rfc3986`



CIViC-Predictive Dataset

GeneName	Variant	chr	start	end	Hgvs_expressions	drug	...
F B	D83 H/Y	1	28592642	28592642	NM_004119.2:c.2503G>C,NP_ 04110.2:p.Asp835His,NM_004119	Sorafenib	
B F	V6 0E		140453136	140453136	NM_004333.4:c.1799T>A,NP_ 04324.2:p.Val600Glu,NC_00000	Cetuximab	
K S	G1 G13	1	25398280	25398285	-	Gefitinib,Erlotinib	
E R	L8 R		55259515	55259515	NC_000007.13:g.55259515T> NM_005228. ...	Erlotinib	
...							

DNA Annotation	Chromosome	~	g.	Location on DNA Sequence	Reference Nucleotide	~	Alternative Nucleotide
CDS Annotation	Gene Name	~	c.	Location on Coding Sequence	Reference Nucleotide	~	Alternative Nucleotide
AA Annotation	Gene Name	~	p.	Reference Amino Acid	Location on Amino Acid Sequence	Alternative Amino Acid	

Figure 7.3: ***GenoConductor***.

functions are **generic** and reusable. Albeit the high trustability and popularity of specific genomic data/knowledge bases, rather than focusing on defining one data operation function for each data/knowledge base, *GenoConductor* supplies a set of generic functions which can be applied as composite functions based on the requirements of different data/knowledge bases. *GenoConductor* is implemented in Python 3.6, publicly available via [19], and compatible to be executed by Dragoman.

7.4 Summary

In this chapter, we show how the diverse interoperability issues existing in textual data and the demand of having a transparent, traceable, and efficient pipeline of KG creation led us to introduce *EABlock*. *EABlock* is an approach to solve entity alignment problems by capturing knowledge from existing KGs while keeping the procedure transparent and traceable. Following the same vision regarding the development of FnO functions which can be made available following the FAIR principles, we introduce *GenoConductor*, a genomic domain-specific library of functions. *GenoConductor* facilitates the alignment and linking between the genomic variation entities by introducing a set of harmonized representations that can be derived from various data sources.

[19]`https://github.com/SDM-TIB/Dragoman/blob/master/Sources/FunctionLibraries/functions_GenoConductor.py`

Chapter 8

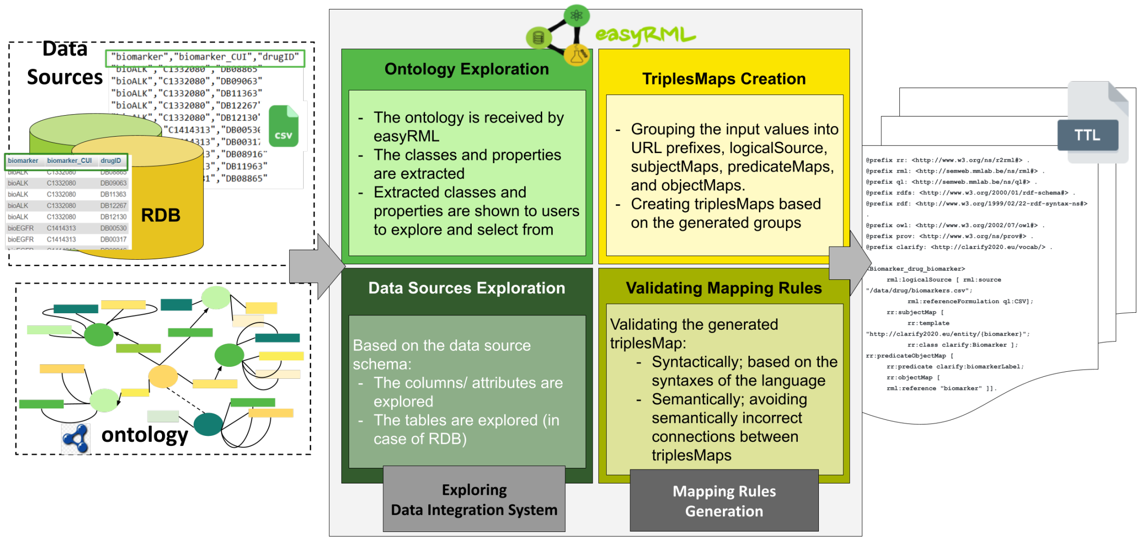
EasyRML

Applying declarative mapping languages such as R2RML and RML provide a transparent data integration system. Materializing such a data integration system ensures the traceability and reproducibility of the process. However, the process of creating mapping rules, the correspondences between the data sources and the concepts in the unified schema can be frustrating, considering the syntaxes of mapping languages. Due to the fact that knowledge engineers who define the mapping rules are experts in the data domain and not necessarily in the semantic web, this process needs to be facilitated. Accordingly, facilitating the process of mapping rules creation, we contribute to the following research question by introducing a tool named *EasyRML*.

RQ5: What are the challenges to applying materialized data integration systems in real-world scenarios?

8.1 *EasyRML*: Creating RML Mapping Rules

To eliminate the complexities caused by syntaxes of mapping languages, we develop and propose *easyRML* which facilitates the creation of declarative mapping rules. Relying on the syntaxes of RML, *easyRML* supports knowledge engineers by allowing them to explore their ontology and data sources while creating the mapping rules to integrate the data sources based on the ontology. *EasyRML* is publicly available: https://labs.tib.eu/sdm/easyrml/.



Ontology Exploration
- The ontology is received by easyRML
- The classes and properties are extracted
- Extracted classes and properties are shown to users to explore and select from

TriplesMaps Creation
- Grouping the input values into URL prefixes, logicalSource, subjectMaps, predicateMaps, and objectMaps.
- Creating triplesMaps based on the generated groups

Data Sources Exploration

Based on the data source schema:
- The columns/ attributes are explored
- The tables are explored (in case of RDB)

Validating Mapping Rules

Validating the generated triplesMap:
- Syntactically; based on the syntaxes of the language
- Semantically; avoiding semantically incorrect connections between triplesMaps

```
@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 rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
.
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix clarify: <http://clarify2020.eu/vocab/> .

<Biomarker_drug_biomarker>
     rml:logicalSource [ rml:source
"/data/drug/biomarkers.csv";
           rml:referenceFormulation ql:CSV];
     rr:subjectMap [
           rr:template
"http://clarify2020.eu/entity/{biomarker}";
           rr:class clarify:Biomarker ];
rr:predicateObjectMap [
     rr:predicate clarify:biomarkerLabel;
     rr:objectMap [
     rml:reference "biomarker" ]].
```

Figure 8.1: **The architecture of *easyRML***. The two steps include the exploration of data integration system and the generation of mapping rules.

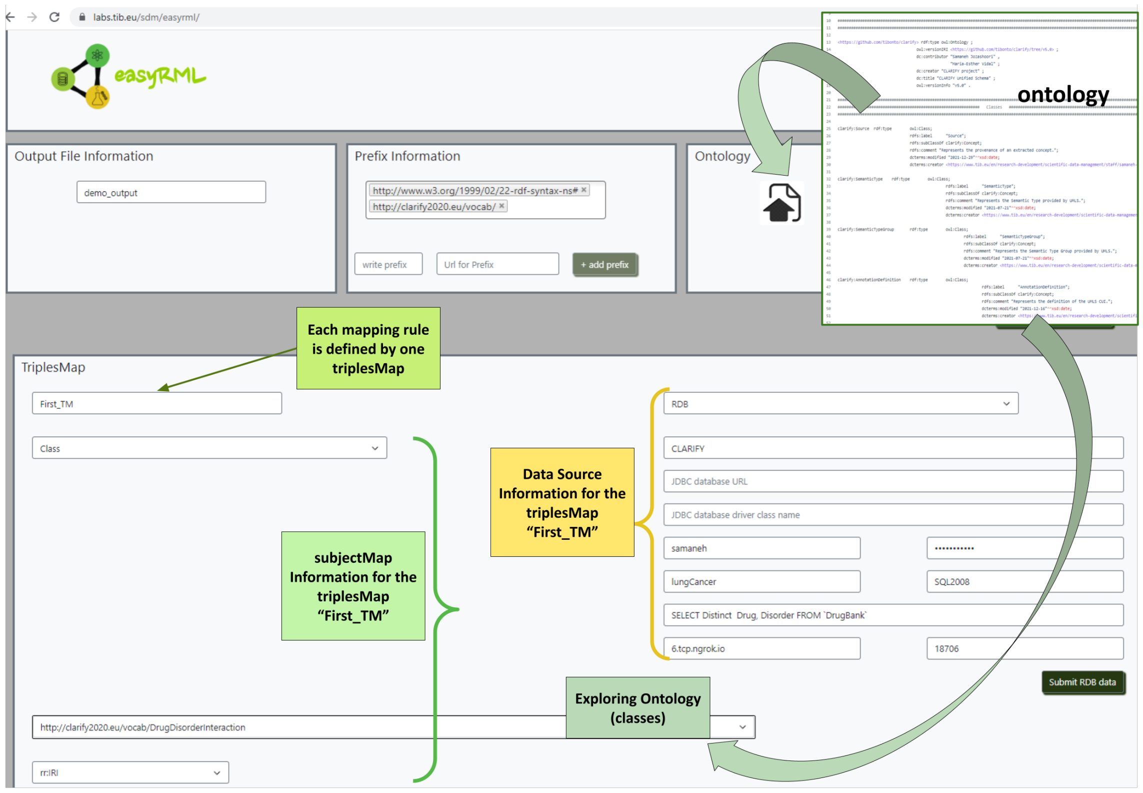
8.2 Architecture and Development

EasyRML aims to facilitate the creation of declarative mapping rules. *EasyRML* provides a platform where users upload the two components of their data integration systems, i.e., data sources [schema] and an ontology, and define the mapping rules to integrate data sources considering the ontology. *EasyRML* is publicly available [20] and maintained by the Scientific Data Management (SDM) group at the TIB – Leibniz Information Center for Science and Technology in Hannover[21]. In addition, *easyRML* is available via GitHub[22] and Docker hub [23].

8.2.1 Exploring Data Integration System

As shown in the Figure 8.1, *easyRML* receives the ontology and the header of the tabular data sources from users. Facilitating the exploration of the ontology and data sources for users, *easyRML* provide classes, properties, and the data attributes to be explored in separated steps where they are required. Correspondingly, classes of the

[20] https://labs.tib.eu/sdm/easyrml/
[21] https://www.tib.eu/en/research-development/scientific-data-management/
[22] https://github.com/SDM-TIB/easyRML
[23] https://hub.docker.com/r/sdmtib/easyrml


labs.tib.eu/sdm/easyrml/
easyRML
ontology
Output File Information
Prefix Information
Ontology
write prefix
Url for Prefix
+ add prefix
Each mapping rule is defined by one triplesMap
TriplesMap
Data Source Information for the triplesMap "First_TM"
subjectMap Information for the triplesMap "First_TM"
JDBC database URL
JDBC database driver class name
Submit RDB data
Exploring Ontology (classes)

demo_output

http://www.w3.org/1999/02/22-rdf-syntax-ns#
http://clarify2020.eu/vocab/

First_TM

Class

RDB

CLARIFY

samaneh

lungCancer

SQL2008

SELECT Distinct Drug, Disorder FROM `DrugBank`

6.tcp.ngrok.io

18706

http://clarify2020.eu/vocab/DrugDisorderInteraction

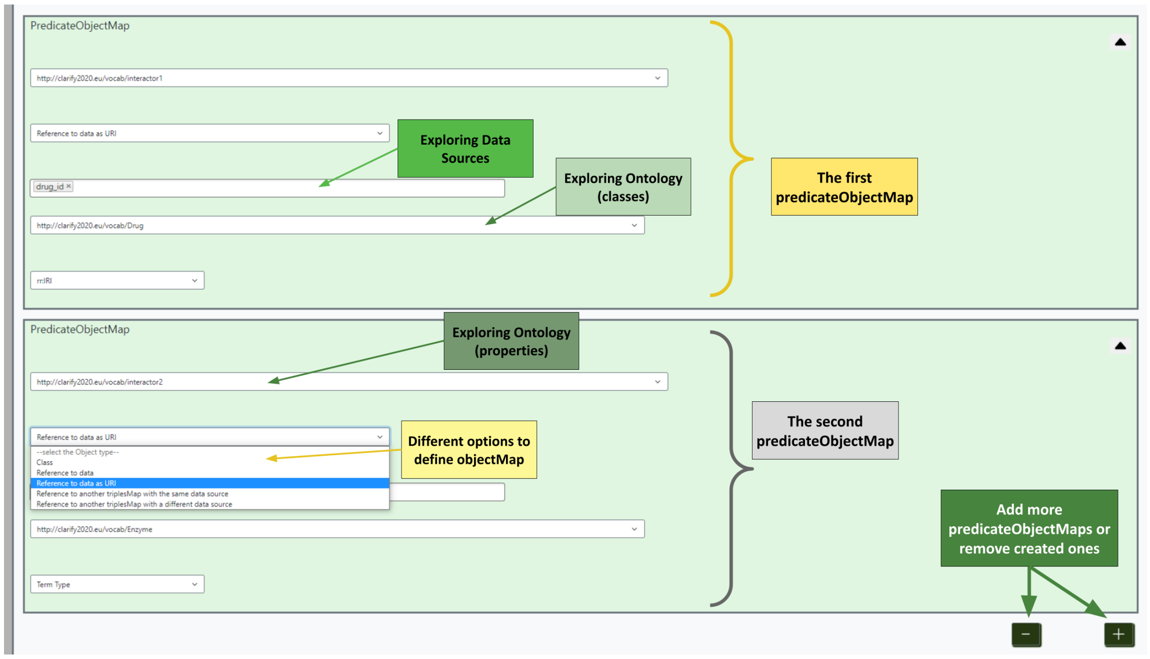
rr:IRI

Figure 8.2: Providing the ontology and data sources to allow for the exploration. Further information about data sources are provided. The subject of the triple is defined.

ontology and data attributes can be explored while defining subjects and objects of triples while the properties are explored during the definition of predicates.

8.2.2 Mapping Rules Generation

Once users finish defining the mappings between their data sources and their ontology, *easyRML* create the `triplesMaps` following the correct structure and syntaxes of the [R2]RML. It should be noted that *easyRML* prevent users to define semantically incorrect rules. For instance, defining the object of a triple based on the subject of another triple that is not previously defined, is not allowed. Therefore, *easyRML* ensures that the generated mapping rules are syntactically and semantically validated.


Exploring Data Sources
Exploring Ontology (classes)
The first predicateObjectMap
Exploring Ontology (properties)
Different options to define objectMap
The second predicateObjectMap
Add more predicateObjectMaps or remove created ones
−
+

PredicateObjectMap

http://clarify2020.eu/vocab/interactor1

Reference to data as URI

drug_id ×

http://clarify2020.eu/vocab/Drug

rr:IRI

PredicateObjectMap

http://clarify2020.eu/vocab/interactor2

Reference to data as URI

--select the Object type--
Class
Reference to data
Reference to data as URI
Reference to another triplesMap with the same data source
Reference to another triplesMap with a different data source

http://clarify2020.eu/vocab/Enzyme

Term Type

Figure 8.3: Defining the predicate and the object of the triple exploring the classes and the properties of the ontology, respectively. The options for defining the object can be also explored. Also, predicate and object can be added or removed.

8.3 Demonstration and Application

Figure 8.2 captures a view of *easyRML* where providing samples of ontology and data sources, one can explore them using *easyRML*. The data source can be provided in two different formats, i.e., Comma-Seperated Values (CSV) or Relational Data Bases (RDB). According to the type of data source, *easyRML* guides them through providing further information about the data source that is required to be added. As it is shown in Figure 8.2 each mapping rule is considered as one block named `triplesMap`, following the RML syntaxes. Having the RDF model in mind, the user one can start defining the first part of a triple, i.e., the subject a.k.a. `subjectMap`. This is the first experience that the user will have exploring the data source schema and the classes of the ontology to define the subject.

The following step is demonstrated in Figure 8.3. In this step, the user further explores the data source schema and the properties and classes of the ontology, as they define the predicates a.k.a. `predicateObjectMap` and the object a.k.a `subjectMap` of the triple. Additionally, they can also explore and notice different options for defining

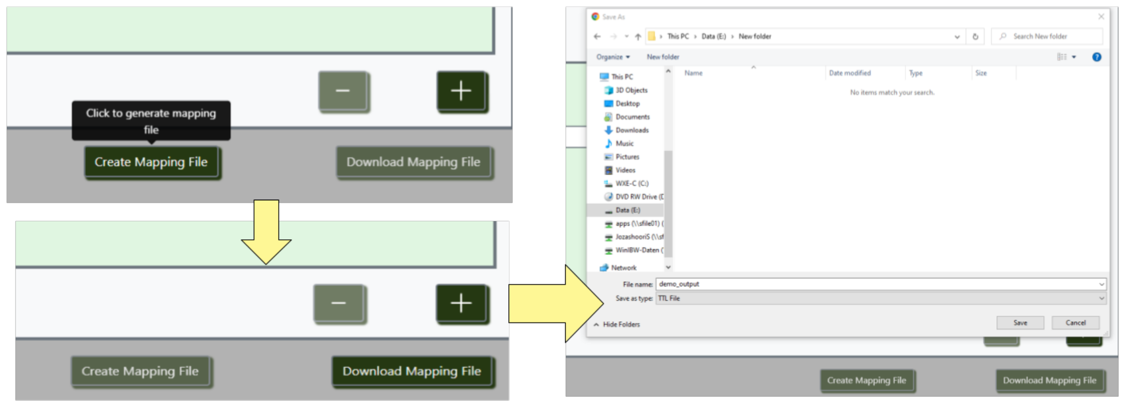


Figure 8.4: demo3...

the object following the RML mapping language. Furthermore, the plus and minus buttons shown in Figure 8.3 allow the user to add or remove `predicateObjectMap` for a specific `triplesMap`.
Lastly, as Figure 8.4 illustrates, the process can be finalized by submitting the data and creating the mapping file using the button named “Create Mapping File”. Then, following the arrows in Figure 8.4, the mapping rules file can downloaded and stored applying the button “Download Mapping File”.

8.4 Summary

In this chapter, we present *easyRML*, a tool to support users creating declarative mapping rules following [R2]RML language. We emphasize that providing a platform to explore the data integration system components in addition to eliminating the syntax complexity layer, can facilitate the process of declarative mapping rules creation for knowledge engineers and different domain experts.

Chapter 9

Dragoman: Mapping-based Optimizations

A challenging step in materializing data integration, is the traceability of procedures that aim to overcome interoperability issues, a.k.a. data-level integration. In most pipelines, data integration is performed by ad-hoc programs, preventing traceability and reusability. Whether data processing or entity alignment, data-level integration can be defined as functions and integrated as part of the mappings performing schema-level integration. However, combining functions with the mappings introduces a new source of complexity that can considerably impact the required number of resources and execution time. In this chapter, we tackle the problem of efficiently executing mappings with functions and formalize the transformation of them into function-free mappings, based on the formalization provided in Chapter 4. Similar to Chapter 5 and Chapter 6, the contributions described in this chapter targets the following research question.

RQ3: How can efficiency be ensured in a materialized data integration system?

The transformations we explain are the basis of an optimization process that aims to perform an eager evaluation of function-based mapping rules. As a result, each function is executed once and efficiently reused. These techniques are implemented in a framework named Dragoman, providing, thus, the possibility to plan the optimized execution of functions and the materialization of reusable functions. The preliminary insight on the contents of this chapter is published in [41] as a collaboration with a Ph.D. student of Polytechnical University of Madrid. The extended and complete contributions of this chapter are under review[24].

[24]https://www.semantic-web-journal.net/content/dragoman-efficiently-evaluating-declarative-mapping-languages-over-framework

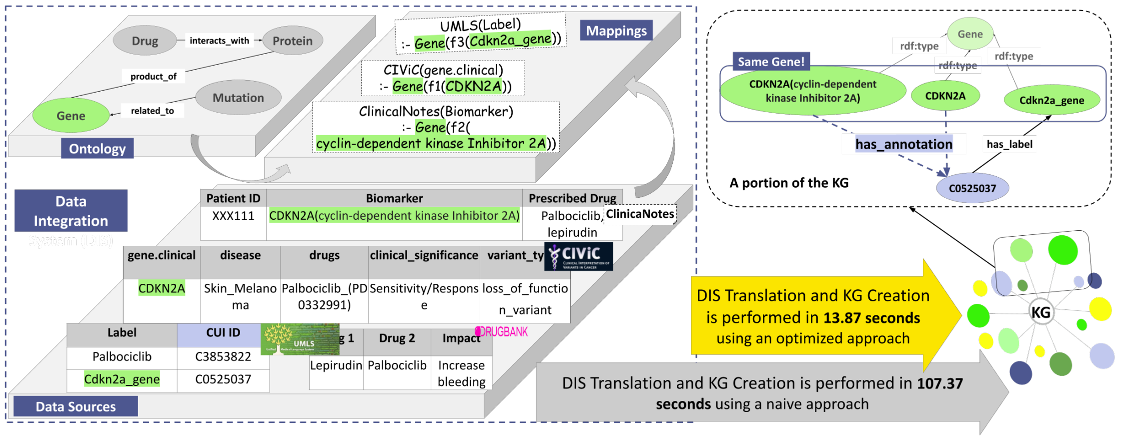

Drug
interacts_with
Protein
product_of
Gene
related_to
Mutation
Ontology
UMLS(Label)
:- Gene(f3(Cdkn2a_gene))
Mappings
CIViC(gene.clinical)
:- Gene(f1(CDKN2A))
ClinicalNotes(Biomarker)
:- Gene(f2(
cyclin-dependent kinase Inhibitor 2A))
Data Integration
Patient ID | Biomarker | Prescribed Drug
XXX111 | CDKN2A(cyclin-dependent kinase Inhibitor 2A) | Palbociclib, lepirudin
ClinicaNotes
gene.clinical | disease | drugs | clinical_significance | variant_ty
CDKN2A | Skin_Melanoma | Palbociclib_(PD0332991) | Sensitivity/Response | loss_of_function_variant
CIViC
Label | CUI ID
Palbociclib | C3853822
Cdkn2a_gene | C0525037
UMLS
Drug 1 | Drug 2 | Impact
Lepirudin | Palbociclib | Increase bleeding
DRUGBANK
Data Sources
Gene
rdf:type
rdf:type
rdf:type
Same Gene!
CDKN2A(cyclin-dependent kinase Inhibitor 2A)
CDKN2A
Cdkn2a_gene
has_annotation
has_label
A portion of the KG
C0525037
KG
DIS Translation and KG Creation is performed in 13.87 seconds using an optimized approach
DIS Translation and KG Creation is performed in 107.37 seconds using a naive approach

Figure 9.1: **Motivating Example**.

9.1 Motivating Example

To better understand the requirement of having a transparent representation of data operations, we explore a real-world example from the biomedical domain that has inspired this work. CDKN2A is one of the critical loci of inactivation at both the germline and somatic mutations in patients with melanoma [68]. Many ongoing studies are established to investigate the mutations related to CDKN2A and its correlation to the diagnosis or prognosis of melanoma, prescribed treatments, interactions between different drugs, and the effectiveness of different treatments in presence of specific mutations. Therefore, a comprehensive insight of actionable knowledge can only be achieved by integrating and semantifying data derived from different studies and residing in various sources. Nevertheless, it is essential to integrate mentioned data in a traceable and transparent manner; traceable, so that the observed results can be explained, and transparent, so as to verify the observed correlations or causation.

Figure 9.1 shows an example of integrating data derived from four different sources i.e., CIViC[25], DrugBank[26], UMLS[27], and clinical notes from hospitals into a knowledge graph. It can be observed that, in addition to the various attribute names that are used to represent gene data in different data sources, the representative values of the same gene i.e., CDKN2A also differ among the sources. Hence, mapping the

[25]https://civicdb.org/home
[26]https://go.drugbank.com/
[27]UMLS

gene values to the "Gene" concept in the ontology only provides the solution to the schema-level integration. The instances of the Gene class need to be aligned performing entity alignment. For this purpose, one solution is to annotate the instances using standard vocabularies or metathesaurus concepts such as Concept Unique Identifier (CUI) from UMLS database [7]; shown in purple in Figure 9.1. The entity alignment task can be defined as a data operation function that accesses an engine performing Named Entity Recognition (NER) and Entity Linking and retrieves the annotation according. Owing to the existing declarative formalism of functions, entity alignment can be part of the main pipeline of the knowledge graph using functions in mappings. Nonetheless, DIS translation as part of the knowledge graph creation - shown in Figure 9.1 - can be a very expensive process in the presence of large heterogeneous data. Therefore, adding another layer i.e., executing data operation functions may increase the complexity. As shown in Figure 9.1, following a naive approach, i.e., no specific optimization for function execution translating a sample DIS including entity alignment functions [42] and 10k of data records, can be 10 times more expensive (in terms of execution time) than an optimized approach. This brings new optimization challenges and the opportunity to overcome them by proposing an approach to scale up the materialized knowledge graph creation in the presence of data operation functions.

9.2 Problem Statement and Proposed Solution

As formally shown in Chapter 4, a knowledge graph corresponds to the evaluation of a data integration system, $DIS_G = \langle O, S, M, F \rangle$ in an interpretation structure I. The mapping assertions in M state the definition of the concepts in the ontology O in terms of the data source signatures in S. Following existing W3C standards, these mapping assertions can be specified in R2RML and RML. F is a set of built-in and user-defined functions for data operations, presented declaratively applying languages such as FnO+RML. The work represented in this chapter tackles the problem of evaluating the user-defined functions efficiently and optimizing the evaluation of the given DIS. Dragoman, the proposed solution in this paper, interprets and transforms the mapping assertions in M and evaluates the functions in F based on the data in S efficiently. Dragoman focuses on scaling up the function evaluation process and transformation of a provided DIS into an optimized, function-free one, as part of the knowledge graph creation process. The outcome of Dragoman is a transformed data integration system as $DIS_G = \langle O, S', M' \rangle$.

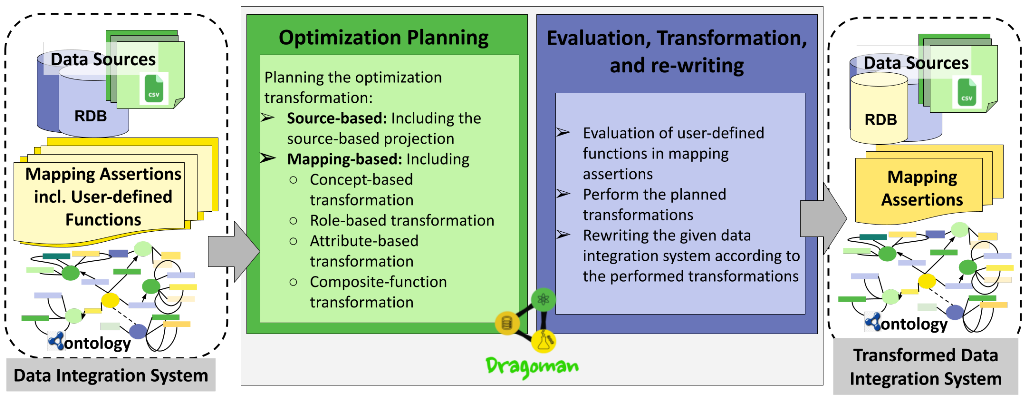

Data Sources
RDB
Mapping Assertions incl. User-defined Functions
ontology
Data Integration System
Optimization Planning
Planning the optimization transformation:
➢ Source-based: Including the source-based projection
➢ Mapping-based: Including
○ Concept-based transformation
○ Role-based transformation
○ Attribute-based transformation
○ Composite-function transformation
Evaluation, Transformation, and re-writing
➢ Evaluation of user-defined functions in mapping assertions
➢ Perform the planned transformations
➢ Rewriting the given data integration system according to the performed transformations
Dragoman
Data Sources
RDB
Mapping Assertions
ontology
Transformed Data Integration System

Figure 9.2: **Approach: Dragoman**.

9.2.1 Problem Statement

Given a data integration system $DIS_G = \langle O, S, M, F \rangle$ which generates the knowledge graph G, the problem of scaling up the process of knowledge graph creation is defined as the problem of identifying a data integration system $DIS'_G = \langle O, S', M', F \rangle$ such that: **i)** The execution time of $eval(m', \mu(\overline{X'}))$ is *minimized* for all $m' \in M'$ and $\overline{X'} \in S'$. **ii)** The RDF knowledge graphs resulting from evaluating the two data integration systems are *equivalent*, i.e., $\Sigma_{i=1}^{n} eval(m_i, \mu(\overline{X_i})) \equiv \Sigma_{j=1}^{n'} eval({m'}_j, \mu(\overline{X'_j}))$.

9.2.2 Proposed Solution

We propose Dragoman, to efficiently evaluating <u>d</u>ecla<u>ra</u>tive mappin<u>g</u> languages <u>o</u>ver fra<u>m</u>eworks for knowledge gr<u>a</u>ph creatio<u>n</u>. Dragoman framework introduces a set of transformation rules to transform a given DIS_G to the DIS'_G such that the execution time required to create the same knowledge graph G from DIS'_G is less than the required time by DIS to produce the same knowledge graph G. As shown in the Figure 9.2, the transformation rules proposed by Dragoman are grouped into source-based and mapping-based categories. Dragoman plans the required transformations based on the mapping assertions in M given by DIS_G. We explain the transformation rules in detail in section 9.3.

Algorithm 9.2 Function Evaluation Algorithm

```
Require: $S_{ma}(\overline{X}_{ma}), g_{ma}(\overline{T}_{ma}), OLD\_DIS.F$
Ensure: $Execute(S_{ma}(\overline{X}_{ma}), g_{ma}(\overline{T}_{ma}))$
 1: $S_g(\overline{X}_g, \overline{T}) \leftarrow \{\}$
 2: if $\overline{T}_{ma} \in OLD\_DIS.F : g_{ma}(\overline{T}_{ma}) = a_{ma}(b_{ma}(\overline{T}'_{ma}))$ then
 3:     $Execute(S_{ma}(\overline{X}_{ma}), g_{ma}(\overline{T}_{ma}),$
 4:     $OLD\_DIS.F)$
 5: else
 6:     $S_g(\overline{X}_g, \overline{T})$ is generated as a new data
 7:     source, such that $\mu(t, \overline{X}_{ma}) \in \overline{X}_g$
 8:     holds for each $t$ in $\overline{T}_{ma}$ and $S_g(\overline{X}_g, \overline{T})$
 9:     belongs to $\sigma_S(S_g)$
10: end if
11: return $S_g(\overline{X}_g, \overline{T})$
```

Algorithm 9.1 Evaluation, Transformation, and re-writing Algorithm

```
Require: $OLD\_DIS = \langle O, S, M, F \rangle$ and $TRs$
Ensure: $NEW\_DIS = \langle O, S', M', F \rangle$
 1: Suppose $OLD\_DIS.\overline{MF}$ is a set of mapping assertions including terms $g_i(.)$ that
    are user-defined functions: $OLD\_DIS.F : \{g_1(.), ..., g_n(.)\}$ and receive a set of terms
    $\overline{T}_i$ as the arguments.
 2: repeat
 3:     Select from $OLD\_DIS.M$ a mapping
 4:     assertion $ma$
 5:     if $ma$ in $OLD\_DIS.\overline{MF}$ then
 6:         $S_g(\overline{X}_g, \overline{T}) =$
 7:         $Execute(S_{ma}(\overline{X}_{ma}), g_{ma}(\overline{T}_{ma}),$
 8:         $OLD\_DIS.F)$
 9:     end if
10:     $(\overline{Tma}, \overline{TS}) \leftarrow Transformation(ma, TRs)$
11:     $M' = (OLD\_DIS.M - \{ma\}) \cup$
12:     $\overline{Tma}$
13:     $S' = (OLD\_DIS.S \cup \overline{TS})$
14:     $NEW\_DIS.M = M'$
15:     $NEW_DIS.S = S'$
16: until $OLD\_DIS == NEW\_DIS$
17: return $NEW\_DIS$
```

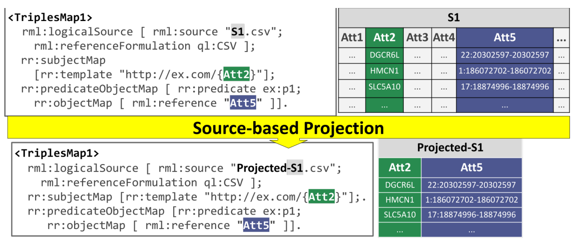
Algorithm 9.1 represents the evaluation, transformation, and re-writing component in Dragoman, also shown in Figure 9.2. As it can be perceived from the Algorithm 9.1, Dragoman relies on an eager evaluation strategy in evaluating user-defined functions in F. Intuitively, the eager evaluation of user-defined functions is the first step in the execution algorithm of Dragoman (lines 1-8). Algorithm 9.2 expresses a sketch of the steps that Dragoman follows in evaluating user-defined functions. To meet the requirements of an eager evaluation, in the case of having composite functions, a(b(.)), the priority of the execution is with the evaluation of functions which are the arguments to the other functions i.e., b(.); Algorithm 9.2 lines 2 and 3. The Eager evaluation strategy enables Dragoman to avoid evaluating the same functions in F with the same arguments more than once, as illustrated in the example in Figure 4.4.

9.3 Transformation Rules

As explained in Algorithm 9.1, lines 9-14, and Algorithm 9.3, after evaluating all the functions in F, Dragoman transforms the mapping assertions in M and data sources in S to the function-free mapping assertions in M' and sources in S'. The key point in optimizing the process of knowledge graph creation is to consider the data sources and mapping assertions coherently [43, 41]. The semantics encoded in the mapping assertions provide insight into the portions of each data source that contribute to the creation of the knowledge graph and the intersections between the data sources. Considering the different types of mapping assertions and user-defined functions, Dragoman introduces five transformation rules as described in the following.

Algorithm 9.3 Transformation Algorithm

```
Require: $ma$, $TRs$
Ensure: $Transformation(ma, TRs)$
 1: $OLD\_\overline{Tma} \leftarrow \{\}$
 2: $NEW\_\overline{Tma} \leftarrow \{\}$
 3: $TS \leftarrow \{\}$
 4: for each $S_i$ in $Sources(ma)$ do
 5:     $S'_i \leftarrow \Pi_{Att_{(ma)}} S_i$
 6:     $ma \leftarrow replace(S_i, S'_i, ma)$
 7:     $\overline{TS} \leftarrow TS \cup \{S'_i\}$
 8: end for
 9: $NEW\_\overline{Tma} \leftarrow \{ma\}$
10: repeat
11:     $OLD\_DIS \leftarrow NEW\_DIS$
12:     switch $TR$ do
13:         case $Concept{-}based\ Transformation$
14:             $(NEW\_\overline{Tma}, \overline{TS}) \leftarrow$
15:             $Concept\_Transformation(ma, \overline{TS})$
16:         case $Role-based\ Transformation$
17:             $(NEW\_\overline{Tma}, \overline{TS}) \leftarrow$
18:             $Role\_Transformation(ma, \overline{TS})$
19:         case $Attribute{-}based\ Transformation$
20:             $(NEW\_\overline{Tma}, \overline{TS}) \leftarrow$
21:             $Attribute\_Transformation(ma, \overline{TS})$
22: until $OLD\_\overline{Tma} = NEW\_\overline{Tma}$
23: return $OLD\_\overline{Tma}, \overline{TS}$
```



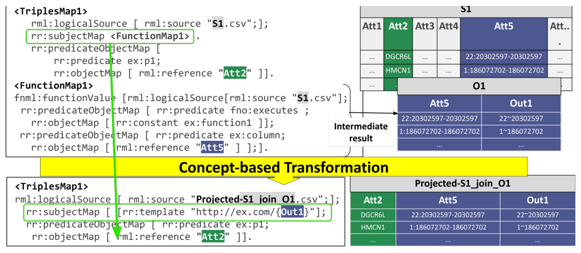
```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
    rml:referenceFormulation ql:CSV ];
  rr:subjectMap
    [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [ rr:predicate ex:p1;
    rr:objectMap [ rml:reference "Att5" ]].
```

S1					
Att1	Att2	Att3	Att4	Att5	...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

```
<TriplesMap1>
  rml:logicalSource [ rml:source "Projected-S1.csv";
    rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];.
  rr:predicateObjectMap [rr:predicate ex:p1;
    rr:objectMap [ rml:reference "Att5" ]].
```

Projected-S1	
Att2	Att5
DGCR6L	22:20302597-20302597
HMCN1	1:186072702-186072702
SLC5A10	17:18874996-18874996
...	...

(a) Source-based Projection



```
<TriplesMap1>
   rml:logicalSource [ rml:source "S1.csv";];
   rr:subjectMap <FunctionMap1> .
   rr:predicateObjectMap [
      rr:predicate ex:p1;
      rr:objectMap [ rml:reference "Att2" ]].
<FunctionMap1>
fnml:functionValue [rml:logicalSource[rml:source "S1.csv"];
 rr:predicateObjectMap [ rr:predicate fno:executes ;
   rr:objectMap [ rr:constant ex:function1 ]];
 rr:predicateObjectMap [ rr:predicate ex:column;
   rr:objectMap [ rml:reference "Att5" ] ];].
```

S1					
Att1	Att2	Att3	Att4	Att5	Att...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	...				

O1	
Att5	Out1
22:20302597-20302597	22~20302597
1:186072702-186072702	1~186072702
...	...

```
<TriplesMap1>
rml:logicalSource [ rml:source "Projected-S1_join_O1.csv";];
  rr:subjectMap [ [rr:template "http://ex.com/{Out1}"];
  rr:predicateObjectMap [ rr:predicate ex:p1;
   rr:objectMap [ rml:reference "Att2" ]].
```

Projected-S1_join_O1		
Att2	Att5	Out1
DGCR6L	22:20302597-20302597	22~20302597
HMCN1	1:186072702-186072702	1~186072702
...	...	...

(b) Concept-based Transformation

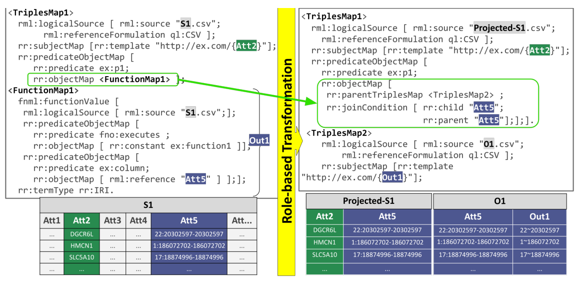
Figure 9.3: **Dragoman Source-based Projection and Concept-based Transformation**

Source-based Projection. The most generic transformation that Dragoman performs before executing the functions on any mapping assertion provided in M is the projection of the attributes in sources of S referred to by the mapping assertion in M. For each mapping assertion in M, independent of the type, Dragoman projects all the attributes needed by that mapping assertion into a new data source and removes the duplicated values. A sketch of this transformation rule is provided in Algorithm 9.3 lines 4-8. It should be noted that if a function in F is referred to by a mapping assertion, then the attributes required are also projected into the new data source. In other words, this transformation pushes down the projection of required attributes and the duplicated values removal [43]. Accordingly, the mapping assertion is transformed to use the new projected data source instead of the original one. To better understand, Figure 9.3a shows an example of this transformation. As it can be seen in Figure 9.3a, from the source $S1$ only two attributes $Att2$ and $Att5$ are utilized in the mapping assertions; $Att2$ as the value for `rr:subjectMap` or the concept mapping assertion, and $Att5$ as the value of the `rr:objectMap` or the single-role mapping assertion. Therefore, the transformed DIS (shown at the bottom of Figure 9.3a) is comprised of the newly generated data source, including only $Att2$ and $Att5$, and the transformed mapping assertion which applies the later data source as the `rr:logicalSource`.

Algorithm 9.4 Concept-based Transformation Algorithm

```
Require: $ma, \overline{TS}$
Ensure: $Concept\_Transformation(ma, \overline{TS})$
 1: $NEW\_\overline{Tma} \leftarrow \{ma\}$
 2: $OLD\_\overline{Tma} \leftarrow NEW\_\overline{Tma}$
 3: suppose $ma : S(\overline{X}) : -C(f(g(\overline{T})))$
 4: where $g$ is a term that is a user-defined
 5: function and receives $\overline{T}$, a set of terms,
 6: as arguments: $\overline{T} : \{t_1, ..., t_m\}$, $\overline{Rma}$
 7: is a set of $Rma_i$ defined as:
 8: $S_i(\overline{X}_{i,1}) : -P(ma, f_2(t_2))$, $\overline{Ama}$
 9: is a set of $Ama_i$ defined as:
10: $S_i(\overline{X}_{i,1}) : -P(ma, t_2)$
11: if $g(.) = a(b(.))$ then
12:     $S_g(\overline{X}_g, b(\overline{T}))$ is defined for all
13:     the values of $b(\overline{T})$ such that
14:     $eval(S_g(\overline{X}_g, b(\overline{T}))) = True$
15:     $ma'$ :
16:     $S(\overline{X}), S_g(\overline{X}_g, b(\overline{T})) : -C(f(t'))$
17:     where $t'$ is a term: $t' \in S_g$
18:     for each $Rma$ : in $\overline{Rma}$ do
19:         $Rma' : S_i(\overline{X}_{i,1})$,
20:         $S_g(\overline{X}_g, b(\overline{T}) : -P(ma', f_2(t_2))$
21:         $\overline{Rma'} \leftarrow \overline{Rma'} \cup Rma'$
22:     end for
23:     for each $Ama$ : in $\overline{Ama}$ do
24:         $Ama' : S_i(\overline{X}_{i,1})$,
25:         $S_g(\overline{X}_g, b(\overline{T}) : -P(ma', t_2)$
26:         $\overline{Ama'} \leftarrow \overline{Ama'} \cup Ama'$
27:     end for
28: else
29:     $S_g(\overline{X}_g, \overline{T})$ is defined for all the
30:     values of $t_i \in \overline{T}, 0 < t_i < m + 1$
31:     such that $eval(S_g(\overline{X}_g, \overline{T})) = True$
32:     $ma'$ :
33:     $S(\overline{X}), S_g(\overline{X}_g, \overline{T}) : -C(f(t'))$
34:     where $t'$ is a term: $t' \in S_g$
35:     for each $Rma$ : in $\overline{Rma}$ do
36:         $Rma' : S_i(\overline{X}_{i,1})$,
37:         $S_g(\overline{X}_g, \overline{T}) : -P(ma', f_2(t_2))$
38:         $\overline{Rma'} \leftarrow \overline{Rma'} \cup Rma'$
39:     end for
40:     for each $Ama$ : in $\overline{Ama}$ do
41:         $Ama' : S_i(\overline{X}_{i,1})$,
42:         $S_g(\overline{X}_g, \overline{T}) : -P(ma', t_2)$
43:         $\overline{Ama'} \leftarrow \overline{Ama'} \cup Ama'$
44:     end for
45: end if
46: $NEW\_\overline{Tma} \leftarrow \{ma'\} \cup \overline{Rma} \cup \overline{Ama}$
47: $\overline{TS} \leftarrow \overline{TS} \cup S_g$
48: return $NEW\_\overline{Tma}, \overline{TS}$
```

Concept-based Transformation. When the output of user-defined functions (`fnml:FunctionTermMap`) is applied as the value of a concept mapping assertion or `rr:subjectMap`, the concept-based transformation is performed on the given mapping assertions in M and sources in S. Figure 9.3b shows an example where the transformation of sources in S is performed by joining the data sources consisting of the function's output, i.e., "intermediate result", and the outcome of the source-based projection on S. After the generation of the new data source, Dragoman, transforms the mapping assertion so that the old data source is replaced with the newly generated one, as shown at the bottom of Figure 9.3b. Algorithm 9.4 provides a sketch of the concept-based Transformation rule and how to implement it. Algorithm 9.4 receives a set of mapping assertions as input. These mapping assertions fulfilling the definitions in lines 3-10. Following an eager evaluation, the Algorithm 9.4 first evaluates the most inner functions in case of having composite functions, i.e., b(.) in the case of a(b(.)), shown in lines 11-28. At the same time, the algorithm performs the concept-based transformation and creates the transformed mapping assertions as shown in lines 12-15. When the Algorithm 9.4 reaches a simple function or the most outer function in a composite function, i.e., a(.), it evaluates it as shown in 16-18 and executes the concept-based transformations, lines 19-22. Finally, the new set of mapping assertions including the transformed mapping assertions is returned.



```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
      rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
    rr:predicate ex:p1;
    rr:objectMap <FunctionMap1> ];
<FunctionMap1>
  fnml:functionValue [
   rml:logicalSource [ rml:source "S1.csv";];
   rr:predicateObjectMap [
     rr:predicate fno:executes ;
     rr:objectMap [ rr:constant ex:function1 ]];
   rr:predicateObjectMap [
     rr:predicate ex:column;
     rr:objectMap [ rml:reference "Att5" ] ];];
  rr:termType rr:IRI.
```

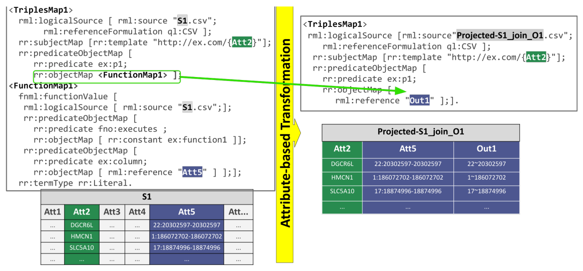
S1					
Att1	Att2	Att3	Att4	Att5	Att...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

```
<TriplesMap1>
  rml:logicalSource [ rml:source "Projected-S1.csv";
    rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
    rr:predicate ex:p1;
    rr:objectMap [
     rr:parentTriplesMap <TriplesMap2> ;
     rr:joinCondition [ rr:child "Att5";
                        rr:parent "Att5"];];].
 <TriplesMap2>
    rml:logicalSource [ rml:source "O1.csv";
        rml:referenceFormulation ql:CSV ];
    rr:subjectMap [rr:template
"http://ex.com/{Out1}"];
```

Projected-S1	
Att2	Att5
DGCR6L	22:20302597-20302597
HMCN1	1:186072702-186072702
SLC5A10	17:18874996-18874996
...	...

O1	
Att5	Out1
22:20302597-20302597	22~20302597
1:186072702-186072702	1~186072702
17:18874996-18874996	17~18874996
...	...

(a) Role-based Transformation



```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
      rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
    rr:predicate ex:p1;
    rr:objectMap <FunctionMap1> ];
<FunctionMap1>
  fnml:functionValue [
   rml:logicalSource [ rml:source "S1.csv";];
   rr:predicateObjectMap [
     rr:predicate fno:executes ;
     rr:objectMap [ rr:constant ex:function1 ]];
   rr:predicateObjectMap [
     rr:predicate ex:column;
     rr:objectMap [ rml:reference "Att5" ] ];];
  rr:termType rr:Literal.
```

S1					
Att1	Att2	Att3	Att4	Att5	Att...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

```
<TriplesMap1>
 rml:logicalSource [rml:source"Projected-S1_join_O1.csv";
   rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
   rr:predicate ex:p1;
   rr:objectMap [
     rml:reference "Out1" ];].
```

Projected-S1_join_O1		
Att2	Att5	Out1
DGCR6L	22:20302597-20302597	22~20302597
HMCN1	1:186072702-186072702	1~186072702
SLC5A10	17:18874996-18874996	17~18874996
...	...	...

(b) Attribute-based Transformation

Figure 9.4: **Dragoman Role-based and Attribute-based Transformations**

Role-based Transformation. Contrary to the previous transformation, role-based transformation is performed once the output of functions (`fnml:FunctionTermMap`) are utilized to build the value of a role mapping assertion (`rr:objectMap`). In contrast to concept-based transformation, role-based transformation forces the join between the data sources consisting of the output of the function and the outcome of the source-based projection, to the mapping assertions. In other words, the output of the function evaluation is stored in a separate new data source and the newly generated data source by the source-based projection. Consequently, the role mapping assertion is transformed to multi-sources role mapping assertions to replace the old data source with two data sources, i.e., the data source, including the output of the functions and the data source generated as the result of the source-based projection. Figure Figure 9.4a illustrates an example of role-based transformation. As shown in the right-hand side of the figure Figure 9.4a, the transformed data sources include two newly generated data sources, and the role mapping assertion is transformed to a multi-sources mapping assertion, applying `joinCondition`. An outline of the implementation of the role-based transformation rule is provided in Algorithm 9.5. The Algorithm 9.5 receives a set of mapping assertions following the assumptions of lines 3-7. Similar to the algorithm of concept-based transformation, Algorithm 9.5 also prioritizes the evaluation of the most inner functions to the outer functions, in the case of composite functions. As expected from the name of the algorithm, Algorithm 9.5 differs from the previous algorithm in the transformation rule. After evaluating the functions, Algorithm 9.5 performs role-based transformation, shown in lines 12-15 and 19-22. The outcome of this algorithm is the updated set of mapping assertions including previously added mapping assertions and the ones transformed according to this algorithm.

Algorithm 9.5 Role-based Transformation Algorithm

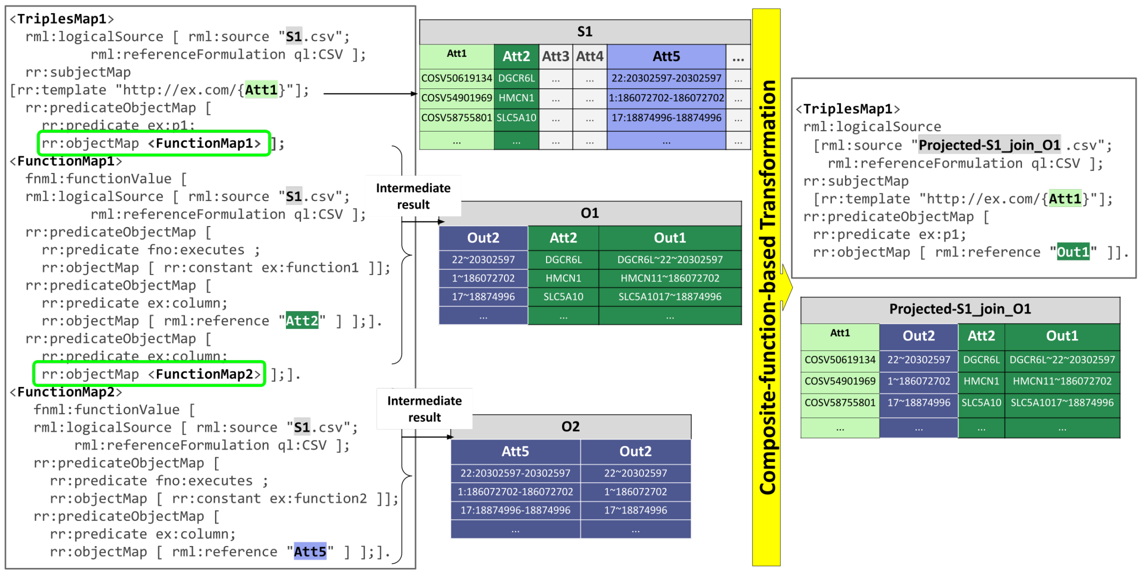
```
Require: $ma, \overline{TS}$
Ensure: $Role\_Transformation(ma, \overline{TS})$
 1: $NEW\_\overline{Tma} \leftarrow \{ma\}$
 2: $OLD\_\overline{Tma} \leftarrow NEW\_\overline{Tma}$
 3: Suppose $ma$ :
 4: $S_i(\overline{X}_{i,1}) : -P(f_1(t_1), f_2(g(\overline{T}_2)))$
 5: where $g$ is a *term* that is a user-defined
 6: function and receives $\overline{T}_2$, a set of *terms*
 7: as arguments: $\overline{T}_2 : \{t_{2,1}, ..., t_{2,m}\}$
 8: if $g(.) = a(b(.))$ then
 9:     $S_g(\overline{X}_g, b(\overline{T}_2))$ is defined for all the values of
10:     $b(\overline{T}_2)$ such that $eval(S_g(\overline{X}_g, b(\overline{T}_2)))$
11:     $= True$
12:     $ma' : S_i(\overline{X}_{i,1}), S_g^{Cma'}(\overline{X}_g, b(\overline{T}_2)),$
13:     $\theta_g(\overline{X}_i, 1, \overline{X}_g) : -P(f_1(t_1), f_2(t'_2))$
14:     where $t'_2$ is a term: $t' \in S_g$
15:     $Cma' : S_g(\overline{X}_g, b(\overline{T}_2)) : -C'(f_2(t'_2))$
16: else
17:     $S_g(\overline{X}_g, \overline{T}_2)$ is defined for all the values of $\overline{T}_2$
18:     such that $eval(S_g(\overline{X}_g, \overline{T}_2)) = True$
19:     $ma' : S_i(\overline{X}_{i,1}), S_g^{Cma'}(\overline{X}_g, \overline{T}_2),$
20:     $\theta_g(\overline{X}_i, 1, \overline{X}_g) : -P(f_1(t_1), f_2(t'_2))$
21:     where $t'_2$ is a term: $t' \in S_g$
22:     $Cma' : S_g(\overline{X}_g, \overline{T}_2) : -C'(f_2(t'_2))$
23: end if
24: $NEW\_\overline{Tma} \leftarrow \{ma', Cma'\}$
25: $\overline{TS} \leftarrow \overline{TS} \cup \{S_g\}$
26: return $NEW\_\overline{Tma}, \overline{TS}$
```

Attribute-based Transformation. This transformation is required when the output of the user-defined function is used as the *term* value of an Attribute mapping assertion. Similar to the concept-based transformation, data sources are transformed by joining the data sources consisting of the output of the function and the result of the source-based projection on S. Consequently, the mapping assertion is transformed to include the newly generated joined data source as the replacement for the original data source. As it can be observed in the example shown in figure Figure 9.4b using the output of user-defined functions - `fnml:FunctionTermMap`- as the *term* value of both role and attribute mapping assertions are very similar in [R2]RML.; they are differentiated by the values `rr:IRI` and `rr:Literal` for `termType` in case of role and attribute mapping assertions, respectively. The attribute-based transformation rule can be implemented following Algorithm 9.6. The Algorithm 9.6 follows the same structure as Algorithm 9.4 and Algorithm 9.5. However, Algorithm 9.6 performs attribute-based transformation which is provided in lines 7-9 and 13-15.

Algorithm 9.6 Attribute-based Transformation Algorithm

```
Require: $ma, \overline{TS}$
Ensure: $Attribute\_Transformation(ma, \overline{TS})$
1:  $NEW\_\overline{Tma} \leftarrow \{ma\}$
2:  $OLD\_\overline{Tma} \leftarrow NEW\_\overline{Tma}$
3:  Suppose $ma : S_1(\overline{X}_{i,1}) : -A(f(t_1), g(\overline{T}_2))$ and $Cma : S_1(\overline{X}_{1,1}) : -C(F_1(t_1))$ where $g(.)$ is a term that is a user-defined function and receives $\overline{T}_2$, a set of terms as arguments: $\overline{T}_2 : \{t_{2,1}, ..., t_{2,m}\}$
4:  if $g(.) = a(b(.))$ then
5:      $S_g(\overline{X}_g, \overline{T}_2)$ is defined for all the values of $b(\overline{T}_2)$
6:      such that $eval(S_g(\overline{X}_g, b(\overline{T}_2))) = True$
7:      $ma' : S_1(\overline{X}_{i,1}), S_g(\overline{X}_g, \overline{T}_2) : -A(f(t_1), t'_2)$
8:      where $t'_2$ is a term: $t'_2 \in S_g$
9:      $Cma' : S_1(\overline{X}_{1,1}), S_g(\overline{X}_g, b(\overline{T}_2)) : -C(f(t_1))$
10: else
11:     $S_g(\overline{X}_g, \overline{T}_2)$ is defined for all the values of $\overline{T}_2$
12:     such that $eval(S_g(\overline{X}_g, b(\overline{T}_2))) = True$
13:     $ma' : S_1(\overline{X}_{i,1}), S_g(\overline{X}_g, b(\overline{T}_2)) : -A(f(t_1), t'_2)$
14:     where $t'_2$ is a term: $t'_2 \in S_g$
15:     $Cma' : S_1(\overline{X}_{1,1}), S_g(\overline{X}_g, \overline{T}_2) : -C(f(t_1))$
16: end if
17: $NEW\_\overline{Tma} \leftarrow \{ma', Cma'\}$
18: $\overline{TS} \leftarrow \overline{TS} \cup \{S_g\}$
19: return $NEW\_\overline{Tma}, \overline{TS}$
```

Composite-Function-based Transformation. When the user-defined function is a composite function, i.e, the output of a `fnml:FunctionTermMap` is an argument to another `fnml:FunctionTermMap`, the same transformation is performed independent of the type of the mapping assertion that refers to the output of the function. composite-function-based transformation is similar to the concept-based transformation; data sources are transformed by joining the data source generated by the output of all functions involved in the composite function, and the data source result from the source-based projection. Accordingly, Dragoman starts executing the composite function from the inner function, i.e., the simple function. As illustrated in the example shown at the bottom of the Figure9.5, after evaluating the inner function `FunctionMap2`, the materialized join between the output and the input attributes of the inner function is provided. This join data source is given to the outer function `FunctionMap1` as the input. After evaluating the outer function, the composite-function-based transformation is performed along with the source-based projection.



```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
          rml:referenceFormulation ql:CSV ];
  rr:subjectMap
[rr:template "http://ex.com/{Att1}"];
  rr:predicateObjectMap [
    rr:predicate ex:p1;
    rr:objectMap <FunctionMap1> ];
<FunctionMap1>
  fnml:functionValue [
  rml:logicalSource [ rml:source "S1.csv";
          rml:referenceFormulation ql:CSV ];
  rr:predicateObjectMap [
    rr:predicate fno:executes ;
    rr:objectMap [ rr:constant ex:function1 ]];
  rr:predicateObjectMap [
    rr:predicate ex:column;
    rr:objectMap [ rml:reference "Att2" ] ];].
  rr:predicateObjectMap [
    rr:predicate ex:column;
    rr:objectMap <FunctionMap2> ];].
<FunctionMap2>
   fnml:functionValue [
   rml:logicalSource [ rml:source "S1.csv";
        rml:referenceFormulation ql:CSV ];
   rr:predicateObjectMap [
     rr:predicate fno:executes ;
     rr:objectMap [ rr:constant ex:function2 ]];
   rr:predicateObjectMap [
     rr:predicate ex:column;
     rr:objectMap [ rml:reference "Att5" ] ];].
```

S1

Att1	Att2	Att3	Att4	Att5	...
COSV50619134	DGCR6L	...	...	22:20302597-20302597	...
COSV54901969	HMCN1	...	...	1:186072702-186072702	...
COSV58755801	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

O1

Out2	Att2	Out1
22~20302597	DGCR6L	DGCR6L~22~20302597
1~186072702	HMCN1	HMCN11~186072702
17~18874996	SLC5A10	SLC5A1017~18874996
...	...	...

O2

Att5	Out2
22:20302597-20302597	22~20302597
1:186072702-186072702	1~186072702
17:18874996-18874996	17~18874996
...	...

```
<TriplesMap1>
 rml:logicalSource
  [rml:source "Projected-S1_join_O1 .csv";
    rml:referenceFormulation ql:CSV ];
 rr:subjectMap
  [rr:template "http://ex.com/{Att1}"];
 rr:predicateObjectMap [
  rr:predicate ex:p1;
  rr:objectMap [ rml:reference "Out1" ]].
```

Projected-S1_join_O1

Att1	Out2	Att2	Out1
COSV50619134	22~20302597	DGCR6L	DGCR6L~22~20302597
COSV54901969	1~186072702	HMCN1	HMCN11~186072702
COSV58755801	17~18874996	SLC5A10	SLC5A1017~18874996
...	...	...	...

Figure 9.5: **Composite-function-based Transformation**.

9.4 Summary

In this chapter, the problem of efficiently creating a knowledge graph from a function-included data integration system is tackled. Dragoman, a system-agnostic engine, is proposed as a solution for optimization and function execution; Dragoman introduces a set of transformations. Relying on an eager evaluation, Dragoman materializes functions in mappings before deciding on the set of required transformations based on provided mapping assertions. Dragoman determines which transformations are needed to be performed on a given data integration system such that the knowledge graph generated by the transferred data integration system is the same as the knowledge graph generated by the original one, nonetheless, in less execution time.

Part III

Empirical Evaluation of Materialized Data Integration Techniques

Chapter 10

SDM-Genomic Testbeds

In recent years materialized creation of knowledge graphs from enormous heterogeneous available data sources on the Web has gained attention. Accordingly, different techniques, frameworks, and engines are introduced to perform different tasks of materialized knowledge graph creation pipelines. Considering the complexity dimensions brought by big data, these techniques, frameworks, and engines may fail to scale up in many cases. Nevertheless, the rapid evolution of data demands scaled-up approaches for frequent materialization and updating of knowledge graphs. For this purpose, different parameters impacting knowledge graph creation need to be studied and corresponding testbeds evaluating the performance of existing approaches are to be provided. In this chapter, we focus on the following research question:

RQ4: Which are the impacting parameters that testbeds need to include to evaluate the advantages of applying enhancement techniques in materialized data integration?

To answer the mentioned research question, first, the parameters that impact the performance of different tasks in a knowledge graph creation pipeline reported by the community [12, 3] or derived by our experimental studies [43, 36, 41] are explained. Then, the testbeds that we have provided to assess each parameter are described. The testbeds discussed in this chapter, *SDM-Genomic Testbeds*, are one of the main contributions of this thesis; they are generated applying biomedical data and real-world scenarios in this domain. The contributions of this chapter are partially published in the empirical study sections of our papers [43, 41, 36, 37] and partially under the review in our latest paper[28].

[28]https://www.semantic-web-journal.net/content/dragoman-efficiently-evaluating-declarative-mapping-languages-over-f

10.1 Data Source Parameters

Knowledge graph creation pipelines from Big Data are required to scale up in terms of different dimensions of Big Data e.g., volume and veracity. For this purpose, evaluating such pipelines demand considering different parameters of the data impacting the process. Hence, we study the following three parameters regarding the data. **Par1: data volume.** The first parameter whose impact needs to be studied is the data source size in terms of the number of records that are integrated into the course of the knowledge graph creation pipeline. **Par2: data veracity (duplicates rate).** The quality of the data sources plays an important role in the data integration process. Accordingly, data veracity is a parameter that is required to be studied while evaluating the performances of knowledge graph creation frameworks. The redundancy rate in data is a characteristic of data that impacts the quality of the data sources; the higher the duplicated data rate, the lower the quality of the data source. **Par3: join selectivity.** The level of the selectivity of the join in multi-sources role mapping assertions is an influential parameter that is essential to be studied. It has been observed in different studies that the behavior of knowledge graph creation pipelines fluctuates when the join selectivity of the data is changed [12].

10.2 Mapping Assertion Parameters

There are several parameters regarding the mapping assertions that impact the performance of any knowledge graph creation pipeline that we explain in the following. **Par4: type of role mapping assertions**. As explained in Chapter 4, role mapping assertions can be divided into three types. Considering the definitions of the three types, they differ in their *body* or *head*. The characteristics that separate these three types of role mapping assertions can also impact on the performance of knowledge graph creation frameworks. **Par5: Number of appearances of the same user-defined function.** The number of mapping assertions with the same user-defined functions that receive the same arguments can affect the performance of an engine considering the strategy that the engine follows. In other words, engines behave differently facing the repetition of the same user-defined functions depending on whether they are relying on an eager or a lazy evaluation. **Par6: number of role mapping assertions**. Whether user-defined functions exist in a set of mapping assertions or not, the total number of function-free role mapping assertions can impact the overall performance of a pipeline. **Par7: complex combinations of multi-sources role mapping assertions.** The existence of complex combinations of multi-sources role mapping assertions, i.e., star join or chain join can negatively impact the performance of the tools involved in knowledge graph creation pipelines significantly. Despite the importance of this parameter to be considered while evaluating any pipeline, the available testbeds and benchmarks such as GTFS-Bench-Madrid [13] have not included them.

10.3 Functions Parameters

As mentioned earlier, the parameters concerning data operation functions have not been studied. Hence, we consider the parameters regarding functions that we assume to be important or we have already observed them being impacting in our preliminary studies explained in Chapter 14. **Par8: types of user-defined functions.** As mentioned earlier the impact of the type of user-defined functions has not been fully studied. To this end, we study the impact of the type of user-defined functions, i.e., whether they are bijective or not. **Par9: the complexity of user-defined functions.** Another unclear characteristic of user-defined functions that may impact the performance of tools is the complexity of the functions. We describe the complexity level of the functions based on the number of required input attributes and operations to be performed. **Par10: composite user-defined functions.** The last parameter that requires to be evaluated is the impact of the complexity of the functions in terms of being "simple" or "composite". The formal definitions of simple and composite functions that are provided in Chapter 4, can be simplified in the following. A user-defined function is a composite function when there exists an argument of it that is the output of another function.

10.4 Testbeds

In order to evaluate the performance of knowledge graph creation pipelines while utilizing different frameworks or engines, different testbeds are required to study the relevant parameters. For this purpose, we classify the testbeds into four groups which are explained in detail in the following. An overview of the parameters that can be studied using each testbed is illustrated in Table 10.2.

10.4.1 TestBed 1: intra-source-based

This testbed provides the setups required to evaluate the performance of different values of Par1, Par3, and Par4 in the presence and absence of intra-mapping assertion source-based optimizations. For the sake of simplicity, we explain this testbed as consisting of two groups; the first group aims to enhance the impact of Par1 and Par3 while the second group targets the combinations of Par3 and Par4..

i) The baseline data set for the first group of testbeds in intra-source-based testbed is produced by randomly selecting 19,503,200 records, equal to 312,1MB overall size, from the combination of three different datasets. The three datasets include mutations, drug-resistant mutations, and protein-RNA interaction predictions data; they are collected from the following data providers: **(i)** The datasets related to mutations and drug-resistant mutations are collected from COSMIC[29], an open source database of somatic mutations in human cancer diseases. **(ii)** A dataset defined by Lang et al. [46] at CRG[30], this dataset includes protein-RNA interaction predictions.
In this testbed, we consider four different data volumes generated by randomly selecting 25%, 50%, 75%, and 100% of the records of the baseline dataset. It also involves three different duplicate - veracity- rates including 25%, 50%, and 75% duplicated-free datasets. It should be noted that all selections of data have been performed randomly to avoid any sampling bias.
ii) The baseline dataset of the second group in intra-source-based testbed is generated by collecting different attributes from various publicly available datasets including the GENCODE reference annotation for the human and mouse genomes [28]. In this dataset, a large amount of selected data relates to exon, the sequence represented in the mature RNA whose mutations can directly affect the sequence of a protein [48]. Since there are overlaps between the data in these datasets, there exists a large number of duplicated values, which makes it a good candidate to study Par3. We generate four datasets from the baseline, each two with the same data attributes and volumes but different qualities, i.e., two are duplicate-free while the other one consists of duplicated values. To study both parameters Par3 and Par4 we create multi-sources role mapping assertions including different combinations of the two datasets. In other words, three multi-sources role mapping assertions each include, two duplicated data sources, two duplicate-free sources, and one duplicated and one duplicate-free data source in their *body*, respectively. The empirical study reported in [43], is set up by applying intra-source-based testbed.

10.4.2 TestBed 2: intra-mapping-based

In general, the intra-mapping-based testbeds are generated to evaluate the performance of knowledge graph creation pipelines and frameworks including user-defined

[29]`https://cancer.sanger.ac.uk/cosmic`
[30]`https://www.crg.eu/`

functions. These testbeds are designed such that different parameters from all three categories, i.e., data source, mapping assertion, and functions to evaluated in the presence or the absence of intra-mapping assertion optimizations.

intra-mapping-based testbeds can be described through two categories. The first category focused only on a few parameters including Par3, Par5, and Par10. Nevertheless, the second category covers 9 parameters out of the 11 parameters explained.
i) The baseline of this testbed is generated by randomly selecting 20,000 records from the coding point mutation dataset in COSMIC[31] database. We keep all 39 attributes of the original dataset in the baseline dataset, while only five to seven of them are utilized in the mapping assertions of TestBed 2. In total, four different RML mapping files are generated consisting of one user-defined function and four, six, eight, or ten mapping assertions including the user-defined function. Two user-defined functions are involved in these testbeds which differ in terms of complexity; “simple” and “complex” functions. The “simple” function is defined to receive one input attribute and perform one operation, while the “complex” function receives two input attributes and completes five operations. The empirical study performed in [41] utilizes this category of testbeds.
ii) The baseline of the second category of intra-mapping-based testbeds is created by combining data from two different publicly available data sources to avoid any possible bias that may have been generated by a particular source in the data values. First, a new subset of the COSMIC mutation dataset[32] is generated. This dataset is created by including 38 attributes of the original COSMIC mutation dataset. Second, another dataset is created by randomly selecting records from the UMLS database[33]. The latest dataset is composed of two attributes; label and CUI identifiers. Combining the explained two datasets creates the baseline of the TestBed 4. We create three datasets by randomly selecting 10k, 100k, and 1 million records from the baseline dataset. In this testbed, 17 sets of mapping assertions combining different types and numbers of mapping assertions and user-defined functions considering explained parameters are provided, overall. We explain the details of each setup in this testbed targeting different parameters in the following. **Par1.** We consider data sources of 10,000, 100,000, and 1,000,000 records in CSV format. It should be noted that only a number of attributes in each data source may participate in the pipeline of the knowledge graph creation, i.e., being applied in mapping assertions. The total number of attributes of data sources are 3-20 times the number of attributes that are

[31]`https://cancer.sanger.ac.uk/cosmic` GRCh37, version90, released August 2019

[32]`https://cancer.sanger.ac.uk/cosmicGRCh37,version90,releasedAugust2019`

[33]`https://www.nlm.nih.gov/research/umls/implementation_resources/applications.html`

Table 10.1: Examples of bijective and non-injective surjective user-defined functions.

Function Type	Function Name	Pre-Condition	Post-Condition
Bijective	**reverseString()**	A case-insensitive string	A case-insensitive string that is the exact reverse of the input string
Non-Injective Surjective	**toLower()**	A case-sensitive string	The exact string as the input string in in lower cases

utilized in mapping assertions. **Par2.** Regarding different percentages of join selectivity, for each data size, we prepare data sources with three levels of join selectivity: low or 80% selectivity rate, medium or 50%, and finally, high or 20% selectivity rate. **Par5.** The mapping assertions in these testbeds are composed of two, four, and six repetitions of the same user-defined function with the same input values. **Par6.** We set up two groups of experiments; the first group includes *star join*s in their original mapping assertions consisting of four multi-sources role mapping assertions with the same MJ. The second group is composed of multi-sources role mapping assertions forming a *chain join*. **Par7.** Regarding the type of user-defined functions parameter, two types of functions including non-injective surjective (NonInjSurj) and bijective are considered. The descriptions of the examples of each type are provided in Table 10.1. **Par8.** Last but not least, both "simple" and "composite" functions are considered in these testbeds; the "composite" function is of type $A(B(C(.)))$.

Table 10.2: The parameters included in each testbed.

		testbeds		
Studied Parameters		TestBed 1	TestBed 2	TestBed 3
Data Source	Par1: Data Volume	✗	✗	✗
	Par2: Selectivity		✗	
	Par3: Data Veracity	✗	✗	✗
Mapping Assertion	Par4: Type of role mapping assertion	✗		
	Par5: # of appearances of the same user-defined functions		✗	
	Par6: # of role mapping assertions		✗	
	Par7: Multiple multi-sources role mapping assertions		✗	✗
Function	Par8: Types of user-defined functions		✗	
	Par9: Complexity of user-defined functions		✗	
	Par10: Composite user-defined functions		✗	

10.4.3 TestBed 3: inter-mapping-based

As shown in Table 10.2, the inter-mapping-based testbeds focus on the evaluation of translating different data integration systems with different data volume, veracity, and complex multi-sources mapping assertions to the knowledge graphs. The baseline dataset of TestBed 3 is generated by randomly selecting records from the coding point mutation dataset in COSMIC[34]. Overall, six datasets of three different sizes, i.e., 10K, 100K, and 1M number of records and two different duplicate rates, i.e., 25% and 75% of duplicates are produced. It should be noted that for each duplicate rate, two datasets are generated; one dataset with each duplicated value being repeated 10 times and another dataset with 20 times of appearance of every duplicated value.

The inter-mapping-based testbeds offer nine mapping assertion configurations. For the sake of clarity, we explain the configurations in two categories. The first category includes six configurations.
Conf1: Set of two mapping assertions with one concept and one attribute mapping assertions. **Conf2:** Set of five mapping assertions, including one concept and four attribute mapping assertions. **Conf3:** Set of four mapping assertions consisting of two concepts, one referenced-source role, and one attribute mapping assertions. **Conf4:** Set of nine mapping assertions with five concepts and four referenced-source role mapping assertions. **Conf5:** Set of three mapping assertions comprised of two concepts and one multi-source role mapping assertions. **Conf6:** Set of nine mapping assertions, including five concepts and four multi-source role mapping assertions. We group the aforementioned mapping assertions into a set named **AllTogether**.

The second category of this group of testbeds includes three extra configurations to enable the evaluation of the impact of two other influential parameters on the performance of KG creation frameworks [59]. **Conf7** aims at evaluating the impact of defining the same predicates using different mapping assertions. **Conf8** provides a mapping rule which is connected to five other mapping rules with different logical sources through join, i.e., this mapping assertion is connected via a five-star join with the other five mapping assertions. The last configuration or **Conf9** combines the first two configurations in one testbed. Accordingly, the three configurations can be summarized as the follows. **Conf7**: Set of four mapping assertions with four concepts and two multi-source role mapping assertions. For each pair of mapping assertions, there is a multi-source role mapping assertion. The data sources of one pair of the mapping assertions are a subset of the other pair. Both pairs of mapping assertions share the same predicate. **Conf8**: Set of six mapping assertions with six concepts and five multi-source role mapping assertions. In this set, five child mapping assertions are referring to the same parent mapping assertion. **Conf9**: Set of eight mapping assertions with eight concepts and seven multi-source role mapping

[34]`https://cancer.sanger.ac.uk/cosmic` GRCh37, version90, released August 2019

assertions.
TestBed 3 enables the experimental study that we perform in [36, 37, 38].

10.4.4 TestBed 4: EA-function-based

In contrast to the previous three testbeds, EA-function-based focus on evaluating the effectiveness of knowledge graph creation pipelines that apply entity linking functions. To create these testbeds, we extract data related to drugs (11,293 records), the disorders for which the drugs are prescribed (416 records), and the interactions between the drugs (1,646,836 records) from DrugBank[35] (version 5.1.8). We produce three mock datasets resembling normal clinical notes for cancer patients, including the data related to comorbidities (1,322 records) and prescribed oncological (1,764 records) and non-oncological drugs (1,325 records). To integrate these data at the schema level, we create a unified schema and a set of mapping assertions. We duplicate the set of mapping assertions to create another set, equivalent to the first one, however, including entity alignment functions as the user-defined functions. This way, generating knowledge graphs from two different testbeds can illustrate the effectiveness of any entity alignment function that is studied. TestBed 4 enables one of the experimental studies that we report in [42].

10.5 Summary

This chapter provides an overview of the parameters impacting the materialization of data integration systems and knowledge graph creation. Relying on the parameters reported by the community and our observations while studying empirically we come up with 10 different parameters that can affect the performance of a materialized knowledge graph creation. Following these parameters, we introduce *SDM-Genomic Testbeds*, a set of various testbeds to evaluate different frameworks materializing data integration systems. *SDM-Genomic* is composed of four group of testbeds; each of them aim focusing on a specific task, e.g., data operation function execution.

[35]`https://go.drugbank.com/`

Chapter 11

Evaluating MapSDI

In this chapter, we describe the empirical study conducted to evaluate the performance of *MapSDI*, introduced in Chapter 5. These results reported in this chapter are evidences that the techniques we propose in Chapter 5 are solution to the RQ3. We compare the performance of MapSDI to the traditional framework for knowledge graph creation which we refer to as "T-framework" from now on in this paper. We aim to answer the following questions:

RQ1) Does MapSDI reduce the required time for knowledge graph creation compared to T-framework?
RQ2) How influential is the performance of MapSDI framework, when data volume increases or data quality decreases?
RQ3) Does MapSDI perform efficiently in case of having more complication in mapping rules e.g., join condition?

The experimental studies described in this chapter are published in [43].

11.1 Experiments

Addressing the research questions presented earlier, we set up 51 experiments overall which are explained as two groups of studies in the followings. In these experiments we rely on TestBed 1 which is described in Subsection 10.4.1. **Metrics** Performance is measured in terms of execution time; it is computed as the elapsed time in seconds between the submission of execution of the framework and the generation of all the RDF triples. The `time` command of the Linux operating system is utilized to measure time. The timeout is set to 500 seconds; the results are visualized based on milliseconds.

Table 11.1: **Four instance datasets size**. The size of four datasets applied in experiments group A with the results being shown in Figure 11.1. The values show how the size of datasets are reduced after the two steps of attribute projection and duplicate removal have been applied, as part of the MapSDI framework.

Data Volume	**Original Size (KB)**	**Pre-processed Size (KB)**
25%	59,200	895
50%	117,900	955
75%	176,400	982
100%	235,000	997

Implementations MapSDI and T-framework are compared on SDM-RDFizer[36] and the rmlmapper-java[37]. The MapSDI framework is implemented in Python 3.6.3 and GNU bash 4.4.12(1) jointly. The experiments are executed on an Ubuntu 17.10 (64 bits) machine with Intel Xeon W-2133, CPU 3.6GHz, 1 physical processor; 6 cores, 12 threads and 64 GB RAM.
Experimental Scenarios We perform in overall 51 experiments; divided into two categories as the follow. **Group** A) The first group of experiments are designed to study the impact of the size of input datasets and their quality in terms of redundancy, on required time for semantic enrichment and integration. In order to avoid the experiments being influenced by other variables such as the number of included attributes and mapping rules, in all experiments of this group, the same one concept is utilized; this concept is represented as a different attribute in each dataset. Additionally, to highlight the difference between the performance of two frameworks, a minimal setup consisting of one attribute in each dataset and consequently one RML triple map, are evaluated. Each 12 experiments that are performed based on a separated framework using a different RDFizer, can be divided into four categories based on the data volume: the **25%**, **50%**, **75%**, and **100% volume**; they are produced by randomly selecting 25%, 50%, 75% and 100% of the records in created dataset, respectively. Subsequently, each mentioned category is divided into three subcategories based on data redundancy; from each generated dataset in the volume category, three datasets are produced by cleaning 25%, 50% and 75% of the data from duplicates. It should be noted that all selections of data have been performed randomly to avoid any sampling bias. **Group** B) The second experiment setup is conducted to study the impact of data redundancy on performance of each framework in case of join condition rules inclusion. Following the same objective, the minimum

[36]`https://github.com/SDM-TIB/TIB-RDFizer`
[37]`https://github.com/RMLio/rmlmapper-java`

amount of required attributes are considered. Accordingly, three experiments are performed on joining two datasets: a) No dataset with duplicates removal; b) One dataset being duplicates-free; and c) Both datasets being duplicates-free.

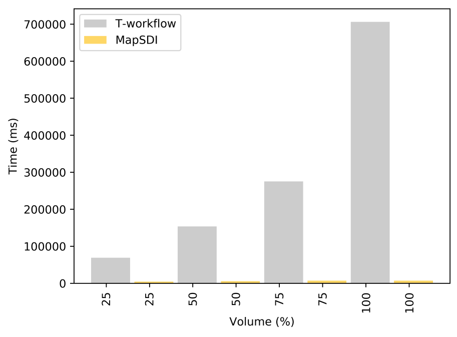
11.1.1 Experimental Results

Answer to RQ1 and RQ2: The results of the experiment group A are shown in Figures 11.1. As it can be observed, MapSDI outperforms T-framework in terms of execution time in all the experiments independently of the RDFizers and percentage of duplicates. This instance of the MapSDI framework performs the *Transformation Rule 3*, i.e., the datasets are merged; while the *Transformation Rule 1* is performed in the two frameworks during the creation of the datasets. According to the results depicted in Figures 11.1, regardless of the RDFizer, the more duplicated data in the datasets, the higher the execution time of the T-framework. It is also important to highlight, the diverse performance ratios of MapSDI and T-framework in terms of the growth of dataset size and data duplicates. MapSDI performs more stable than T-framework. These observations can be explained according to the two steps of pre-processing including attributes projection and duplicate removal that are executed former to the transformation step in the MapSDI framework. The mentioned steps decrease the size of the original datasets considerably. Table 11.1 reports on the reduced size of the input datasets after the pre-processing steps in the experiments conducted over the dataset with 25% data duplicates (Figure 11.1).

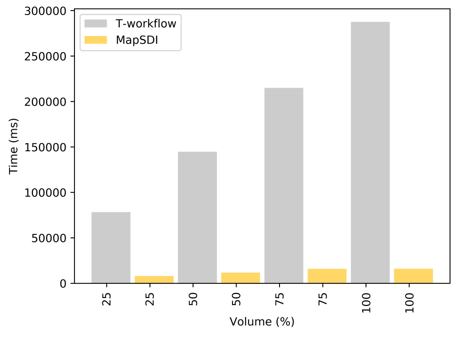
Answer to RQ3: Figure 11.2 illustrates the results of experiments in group B. The rmlmapper timed out in all experiments of group B, the results only refer to the performance of MapSDI and T-framework applying SDB-RDFizer. As it can be observed, the execution time of MapSDI is considerably lower than T-framework in case of having join condition in mapping rules independent of having data duplicates. This instance of MapSDI framework performs the *Transformation Rule 3* as well as *Transformation Rule 2*. The application of these two transformations considerably reduces the number of duplicates and enhances the performance of the SDM-RDFizer during the execution of the join condition between triple maps.

11.2 Summary

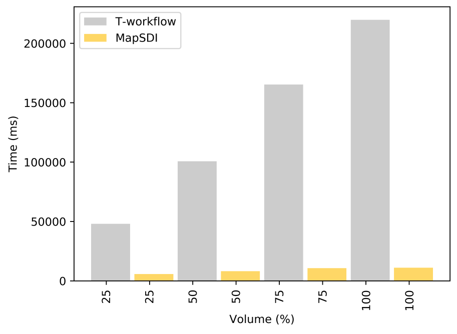
In this chapter, we empirically evaluated the scaled-up framework of semantic enrichment presented in Chapter 5. It was observed from the experiments that the proposed data operation steps in the framework, i.e., attributes projection and duplicate removal, prevent significant fluctuations in the performance of the pipeline while the data is growing. Additionally, the results suggest that the proposed framework leads to execution time savings in the case of joining different data sources.



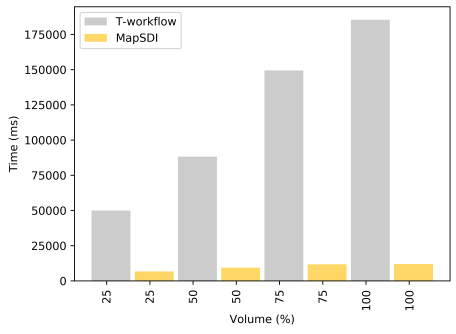
(a) rmlmapper - 75% veracity



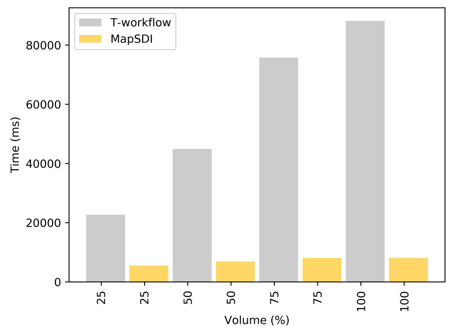
(b) SDM-RDFizer - 75% veracity



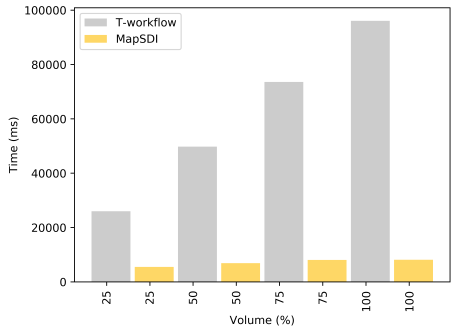
(c) rmlmapper - 50% veracity



(d) SDM-RDFizer - 50% veracity



(e) rmlmapper - 25% veracity



(f) SDM-RDFizer - 25% veracity

Figure 11.1: **Results of experiment group A with different percentage of veracity**. The performance of MapSDI and T-framework on four different sized datasets with 75% redundancy: (a) applying rmlmapper (b) using SDM-RDFizer. MapSDI is able to reduce duplicated and exhibits better performance independently of the data volume and RDFizer. But, the difference between the execution time of two frameworks is much higher when rmlmapper is evaluated.

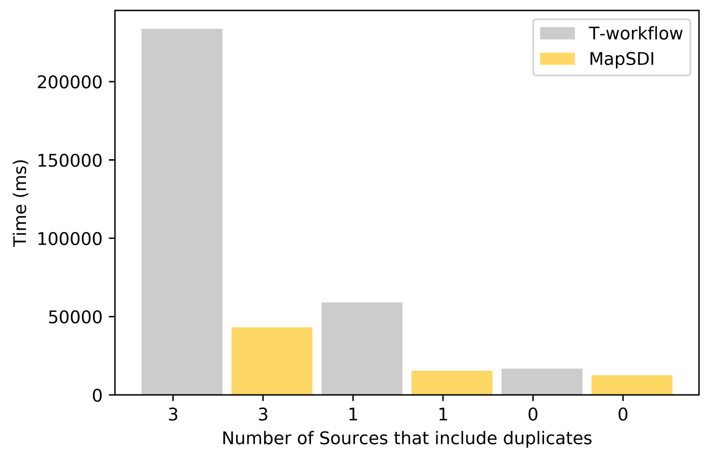


Figure 11.2: **Results of Experiment Group B**. MapSDI and T-framework on two datasets joined by two triple maps. MapSDI performs *Transformation Rule 2 and Rule 3* and it is able to push down projection into the join. With the transformations conducted by MapSDI, the rmlmapper timed out at 500 seconds.

Chapter 12

Evaluating Planner

In this chapter the empirical study performed on the *planner* is reported. The proposed optimization approach is evaluated over state-of-the-art RML-compliant engines, and existing benchmarks of data sources and RML triples maps. Our experimental results suggest that the performance of the studied engines can be considerably improved, particularly in a complex setting with numerous triples maps and large data sources. As a result, engines that time out in complex cases are enabled to produce at least a portion of the knowledge graph by applying the *planner*. These results experimentally prove that the techniques we propose in Chapter 6 answer the RQ3. In this study we rely on the Testbed 4 explained in Subsection 10.4.3. The performance of the solution proposed to the problem of *planning KG creation* is studied in four RML-compliant engines: RMLMapper, RocketRML, SDM-RDFizer, and Morph-KGC. The empirical evaluation aims at answering the following questions:

RQ1) How does planning the execution of mapping assertions affect the performance of the state-of-the-art RML-compliant engines during KG creation?
RQ2) What is the impact of the type of mapping assertions and volume of the data sources on execution time required by engines?

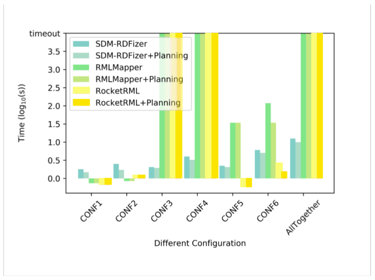
RML Engines. RMLMapper v4.12 [55], RocketRML v1.11.3 [64], Morph-KGC v1.4.1 [3], and SDM-RDFizer v3.6 [60]. Recently, SDM-RDFizer v4.0 [61] has been published. According to the tool description, SDM-RDFizer v4.0 implements planning techniques, physical operators for the execution of mapping assertions, and data compression techniques for reducing the size of the main memory structures required to store intermediate results. In order to create a fair evaluation of the performance of the techniques developed in SDM-RDFizer v4.0, we implement an upgraded version of SDM-RDFizer v3.6 which includes the data compression technique developed in SDM-RDFizer v4.0; we call this engine SDM-RDFizer v4.0^{--}.

Implementations. The planning and execution pipeline is implemented in Python 3. The compression techniques implemented in SDM-RDFizer v4.0^{--} encode RDF resources generated during the KG creation process. For each RDF resource R, an identification number i is assigned to it. Thus, RDF triples are built not from RDF resources but the identification number. Moreover, each identification number i is encoded in Base36 to reduce the memory usage further. Base36 is an encoding scheme that transforms a string into a 36 characters representation. The characters used are the letters from A to Z and the numbers from 0 to 9. For example, the number "95634785" is encoded as "1KXS9T". The SDM-RDFizer operators are adapted to consider this compression method, consuming less main memory.

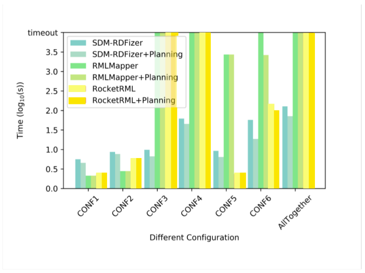
Metrics We consider two metrics to evaluate the efficiency of our proposed approach. *Execution time* is defined as the elapsed time required to generate the bushy tree and execute the corresponding physical plan used to create the KG. It is measured as the absolute wall-clock system time, as reported by the `time` command of the Linux operating system. The leaves of a bushy tree are executed in parallel, and execution of the leaves corresponds to the greatest execution time; execution time also includes the time of merging the results generated during the execution of the tree leaves. *Memory consumption* is determined as the amount of memory that is consumed during the generation of a KG. The memory usage is measured by using the `tracemalloc` library from Python [67]. The `get_traced_memory()` method from `tracemalloc` returns the amount of memory currently being used. This method presents the memory usage in Kilobytes, for ease of use, it is converted into Megabytes. The timeout is five hours. The experiments are executed in an Intel(R) Xeon(R) equipped with a CPU E5-2603 v3 @ 1.60GHz 20 cores, 64GB memory and with the O.S. Ubuntu 16.04LTS.

12.0.1 Experiments - *Efficiency*

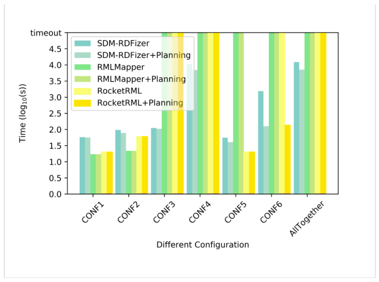
This experiment aims to assess the impact of planning on a real-world dataset provided as the TestBed 4 in Chapter 10. Following the two categories of tesbeds in TestBed 4 described in Chapter 10, we also divide the experiments into two categories; simple and complex mapping assertions.



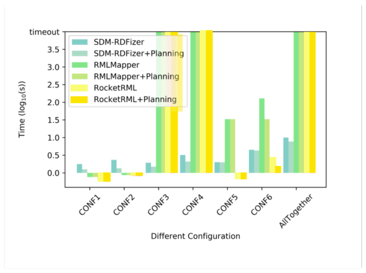
(a) 10k records with 25% duplicate rate.



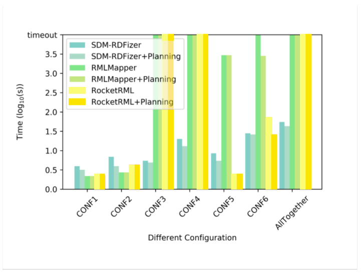
(b) 100k records with 25% duplicate rate.



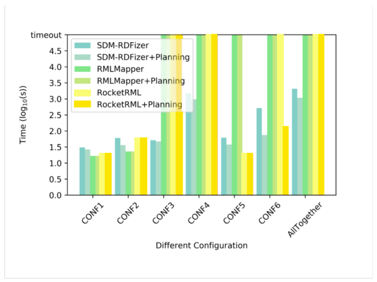
(c) 1M records with 25% duplicate rate.



(d) 10k records with 75% duplicate rate.



(e) 100k records with 75% duplicate rate.



(f) 1M records with 75% duplicate rate.

Figure 12.1: **Results of the Execution of the GENOMIC benchmark.** Execution time of Conf1, Conf2, Conf3, Conf4, Conf5, Conf6, and AllTogether for SDM-RDFizer v3.6, RMLMapper, and RocketRML.

Simple Mapping Assertions

We study the performance of each engine, i.e., RocketRML, RMLMapper, and SDM-RDFizer in presence and absence of planning using SDM-Genomic-Datasets. In addition to the six configurations of mapping assertions, i.e., **Conf1**, **Conf2**, **Conf3**, **Conf4**, **Conf5**, and **Conf6**, we consider an additional configuration consisting of the union of all them. We refer to it as `AllTogether`. As illustrated in Figure 12.1, in the case of having referenced-source role mapping assertions (i.e., **Conf3** and **Conf4**), neither of the two engines, RMLMapper and RocketRML, is able to complete the execution before the timeout. As observed in Figure 12.1, applying planning in simple cases like **Conf1**, **Conf2**, and **Conf3** with low data duplicate rates does not show a considerable impact on the performance.

Conversely, in complex cases such as **Conf6** which include several multi-source role mapping assertions, execution time is reduced significantly exploiting planning. Unfortunately, both RMLMapper and RocketRML lack efficient implementations of the operators that are required to execute referenced-source role mapping assertions. Therefore, the two mentioned engines are unable to finish the execution of **Conf3** and **Conf4** before the timeout (i.e., 5 hours). The results in Figure 12.1 also suggest that with the growth of duplicate data rate, the benefits of using the proposed planning techniques also increased.

Percentage of Duplicates: 25%										
Size	Engine	Conf7			Conf8			Conf9		
		Original	Optimized	% Savings	Original	Optimized	% Savings	Original	Optimized	% Savings
10k	SDM-RDFizer	3.91 sec	5.04 sec	-28.90 %	5.59 sec	6.54 sec	-16.99 %	10.7 sec	6.47 sec	39.53%
	RMLMapper	47.43 sec	36.69 sec	22.64 %	140.27 sec	43.93 sec	68.68 %	180.85 sec	43.25 sec	**76.09 %**
	Morph-KGC	1.81 sec	3.55 sec	-96.13%	1.79 sec	4.22 sec	-135.75 %	2.28 sec	5.2 sec	-128.07 %
100k	SDM-RDFizer	21.14 sec	16.88 sec	20.15 %	99.88 sec	51.11 sec	48.82 %	105.72 sec	44.97 sec	57.46 %
	RMLMapper	3205.37 sec	2628.13 sec	18.01 %	11961.81 sec	3901.14 sec	67.38 %	12593.16 sec	3401.17 sec	**72.99 %**
	Morph-KGC	20.4 sec	19.35 sec	5.14 %	43.87 sec	29.38 sec	33.02 %	42.43 sec	30.84 sec	27.31 %
1M	SDM-RDFizer	177.35 sec	124.08 sec	30.03 %	1656.29 sec	607.06 sec	63.34 %	1769.29 sec	685.22 sec	**61.27 %**
	RMLMapper	TimeOut	TimeOut	-	TimeOut	TimeOut	-	TimeOut	TimeOut	-
	Morph-KGC	1532.94 sec	1224.37 sec	20.13 %	3369.11 sec	2154.92 sec	36.03 %	3329.16 sec	2071.63 sec	37.77 %
Percentage of Duplicates: 75%										
Size	Engine	Conf7			Conf8			Conf9		
		Original	Optimized	%Savings	Original	Optimized	%Savings	Original	Optimized	%Savings
10k	SDM-RDFizer	3.6 sec	4.89 sec	-35.83 %	4.44 sec	5.44 sec	-22.52 %	8.35 sec	5.85 sec	29.94 %
	RMLMapper	38.82 sec	35.41 sec	8.78 %	133.96 sec	47.01 sec	64.90 %	173.08 sec	47.64 sec	**72.47 %**
	Morph-KGC	2.15 sec	4.01 sec	-86.51%	2.11 sec	4.59 sec	-117.53%	2.93 sec	5.33 sec	-81.91%
100k	SDM-RDFizer	19.72 sec	16.16 sec	18.05%	70.5 sec	31.06 sec	55.94%	66.15 sec	29.97 sec	54.69%
	RMLMapper	3203.19 sec	2672.59 sec	16.56%	12669.84 sec	3861.29 sec	69.52%	16541.84 sec	3985.06 sec	**75.90%**
	Morph-KGC	23.53 sec	22.21 sec	5.60%	46.35 sec	35.7 sec	22.97%	48.13 sec	35.68 sec	25.86%
1M	SDM-RDFizer	174.11 sec	123.77 sec	28.91%	983.53 sec	402.59 sec	59.06%	1252.27 sec	516.99 sec	**58.71%**
	RMLMapper	TimeOut	TimeOut	-	TimeOut	TimeOut	-	TimeOut	TimeOut	-
	Morph-KGC	1628.69 sec	1330.01 sec	18.33%	3338.93 sec	2229.78 sec	33.21%	3641.57 sec	2200.08 sec	39.58%

Table 12.1: **SDM-Genomic-Datasets Complex Test Cases**. Duplicate rates are 25% and 75%; Highest Percentage of Savings are highlighted in **bold**. Lowest Percentage of Savings are underlined. The proposed planning and execution techniques are able to enhance the performance of RMLMapper and speed up execution time by up to 76.08%; even in the cases, where RMLMapper timed out, the proposed techniques empower RMLMapper to produce intermediate results. In case of small data sets (e.g., 10K), the proposed techniques may produce overhead in SDM-RDFizer and Morph-KGC (e.g., Conf7 and Conf8).

Complex Mapping Assertions

This experiment aims at assessing the effect of the complex mapping assertions on the execution time during the KG creation process. In these experiments, RocketRML is replaced by Morph-KGC since RocketRML is unable to execute the multi-source mapping assertions that composed the **Conf7**, **Conf8**, and **Conf9**.

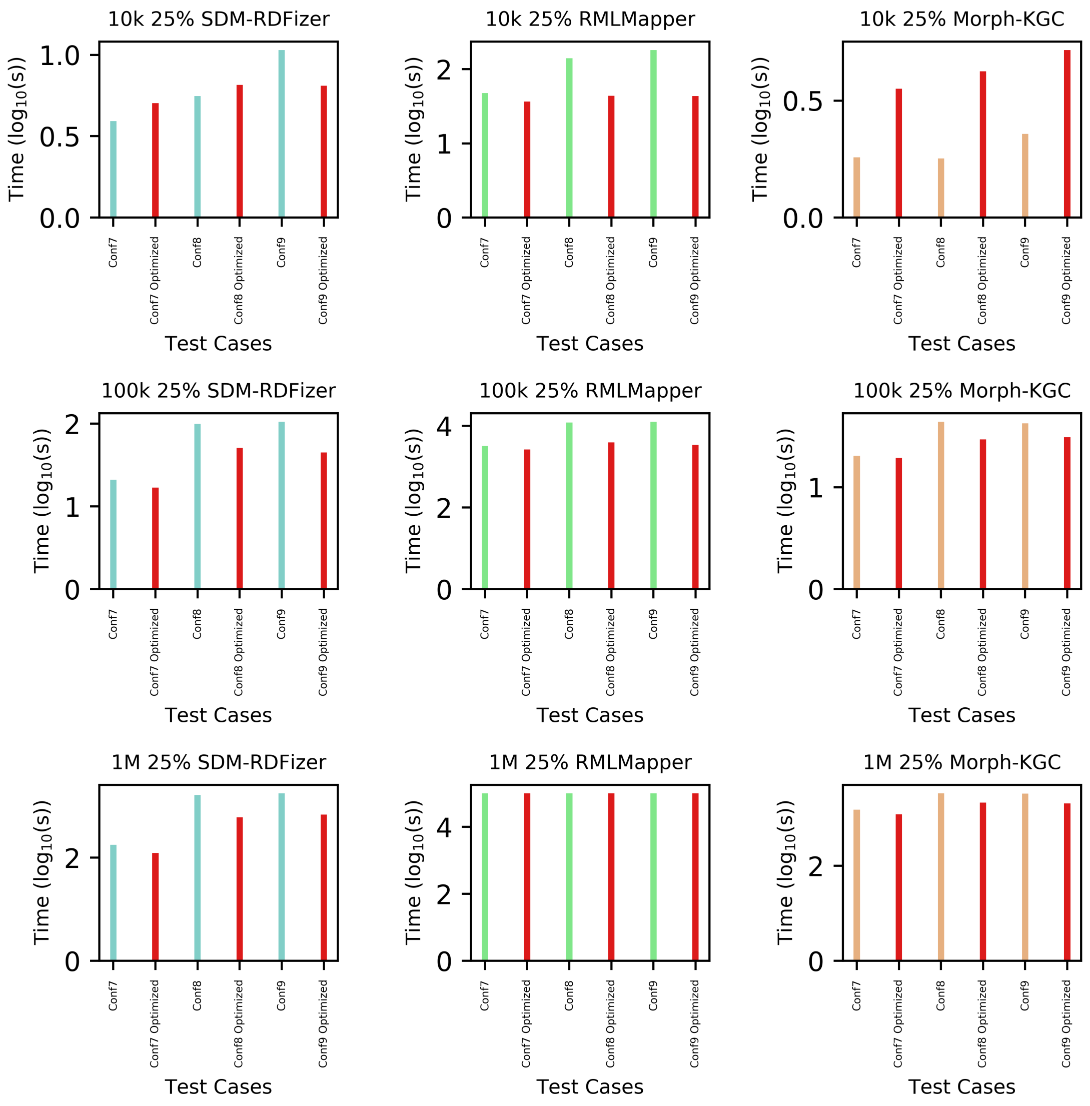


Figure 12.2: **Efficiency Planning For Complex Cases with 25% duplicate rate**. The effects of proposed planning techniques over the SDM-Genomic-Datasets with 25% duplicate rate over Conf7, Conf8, and Conf9. SDM-RDFizer v3.6+Planning, RMLMapper+Planning, Morph-KGC+Planning

Figures 12.2 and 12.3 report on execution time (log scale) and Table 12.1 presents the specific values of each execution. As observed, the RMLMapper performance

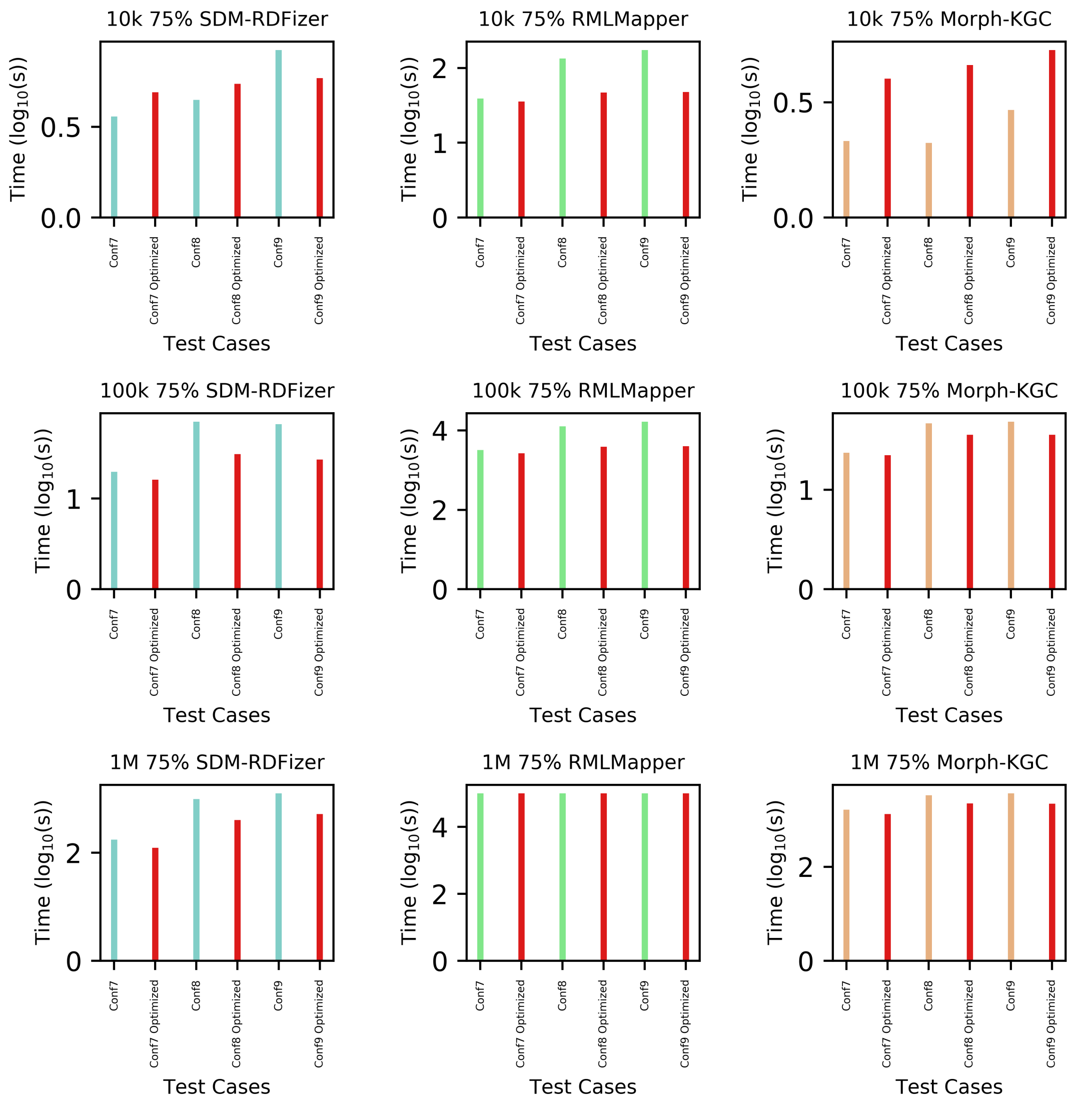


Figure 12.3: **Efficiency Planning For Complex Cases with 75% duplicate rate**. The effects of proposed planning techniques over the SDM-Genomic-Datasets with 75% duplicate rate over Conf7, Conf8, and Conf9. SDM-RDFizer v3.6+Planning, RMLMapper+Planning, Morph-KGC+Planning

is improved in **Conf7**, **Conf8**, and **Conf9** even in data sources of small size, i.e., 10k. In the data source of the size 10k, there is 22.64% reduction of execution time

for **Conf7** with 25% duplicate rate and 8.78% reduction with 75% duplicate rate, 68.68% reduction for **Conf8** with 25% duplicate rate and 64.9% reduction with 75% duplicate rate, and 76.09% reduction for **Conf9** with 25% duplicate rate and 72.47% reduction with 75% duplicate rate. For 100k, there is a 18.01% reduction of execution time for **Conf7** with 25% duplicate rate and 16.56% reduction with 75% duplicate rate, a 67.38% reduction for **Conf8** with 25% duplicate rate and 69.52% reduction with 75% duplicate rate, and a 72.99% reduction for **Conf9** with 25% duplicate rate and 75.90% reduction with 75% duplicate rate.
The RMLMapper timed out after 5 hours with both methods when executing the 1M data sources with all three mappings with duplicate rates. This can be attributed to how the execution of the join is implemented in the RMLMapper and the size of the data. But with the planned execution, it could generate at least a portion of the KG for each mapping. For **Conf7**, **Conf8**, and **Conf9**, respectively, 32.65%, 24.82%, and 28.69% of the KG are generated.
For the SDM-RDFizer and Morph-KGC, there was overhead when generating the KG for **Conf7** and **Conf8** with 10k. This can be attributed to the fact that both the SDM-RDFizer and Morph-KGC already have optimization techniques implemented. Combining the optimization techniques and the physical plan causes the overhead in cases with small data sources, i.e., 10k. While for **Conf9**, there is a 39.53% reduction with 25% duplicate rate and a 29.94% reduction with 75% duplicate rate for the SDM-RDFizer when using the planned execution. There are savings of 100k and 1M when using the planned execution for both engines. In particular, **Conf9** presents the highest savings. For 100k, there is a 57.46% reduction with 25% duplicate rate and a 54.69% reduction with 75% duplicate rate for the SDM-RDFizer and a 27.31% reduction with 25% duplicate rate and a 25.86% reduction with 75% duplicate rate for Morph-KGC.
For 1M, there is a 61.27% reduction with 25% duplicate rate and a 58.71% reduction with 75% duplicate rate for the SDM-RDFizer and a 37.77% reduction with 25% duplicate rate and a 39.58% reduction with 75% duplicate rate for Morph-KGC. This increase in savings is related to the complexity of the mapping; higher complexity causes higher savings.
In conclusion, applying the proposed planning techniques reduces the execution time, independent of the engine by which they are adopted. However, applying these techniques in engines such as SDM-RDFizer and Morph-KGC, which already perform optimization techniques, may cause an overhead. Specifically, in the case of having small size data sources or less complex mapping assertions, the cost of planning in addition to the other optimization techniques implemented in the engine can be higher than the savings. Like any optimization technique, there is a trade-off that

can be estimated based on the provided data integration system. The higher the complexity of the mapping assertions and dataset size, the higher the execution time improvement.

Results Summary

Answer to RQ1. There exist configurations of data integration systems where the proposed planning techniques improve the performance of any state-of-the-art engines. The experimental results provide insights on the cases where planning improves the KG creation frameworks in contrast to the ones that it may cause negative impact. E.g., in case of having small data sources or simple mapping assertions, the execution times of SDM-RDFizer and Morph-KGC are lower ignoring the planning of the mapping assertions. However, it is important to note that execution planning empowers state-of-the-art engines without continuous behavior to generate a partial KG output. In other words, the generated plans enable some engines to produce outputs instead of timing out or running out of memory.

Answer to RQ2. Attribute mapping assertion presents the shortest execution time of all the types of mapping assertion since they represent a simple projection of the raw data. The execution time of a multi-source role mapping assertion depends on the size of the data sources and the number of values associated with them. The execution time of referenced-source role mapping assertions depends on the size of the data source and the data management techniques implemented for each engine. RMLMapper and RocketRML execute the mentioned operation as a Cartesian product, causing the execution time to grow exponentially.

12.1 Summary

In this chapter, we empirically evaluate the execution planning techniques proposed in Chapter 6. The empirical evaluation of the proposed methods empowers existing RML-compliant engines and enables them to scale to complex situations. The results we reported here put into perspective the need for specialized data management methods for scaling up KG creation to complex data integration systems present in real-world applications. Albeit efficiently defined, execution planning may be costly and generate overhead, which negatively impacts engine behavior in simple cases.

Chapter 13

Evaluating EABlock

In this chapter we aim to empirically assess the performance of the *EABlock* which contributed to RQ2. Considering the research question answered in Chapter 7 by proposing *EABlock*, in this chapter we evaluate the performance of our proposed solution. The empirical studies conducted in this chapter are reported in our publication [42]. This experimental study is guided by the following research questions:

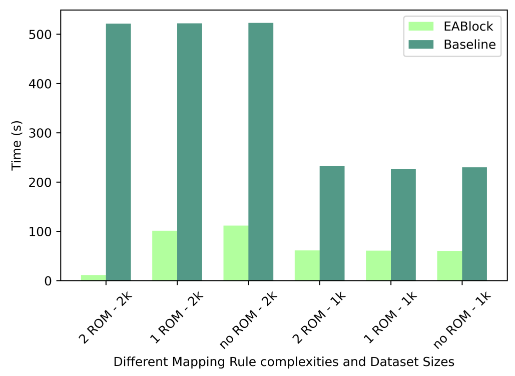
RQ1) What is the impact of applying *EABlock* in KG creation in terms of execution time?
RQ2) How does applying *EABlock* in the process of KG creation impact the quality of the result KG?

Setups. In these experiments, we rely on an API of Falcon[38] for that provides a filtered subset of the background knowledge [57] omitting the resources that are not related to the biomedical domain. A list of related resource types is utilized for filtering the background knowledge. The list contains the following resource types: Chemicals & Drugs, Anatomy, Disorders, Living Beings, Organizations, Physiology, and Genes & Molecular Sequences. Applying this filtering to the background knowledge of Falcon reduces the ambiguity among the resources in the EL task and clears the noise that can be generated by irrelevant resources.

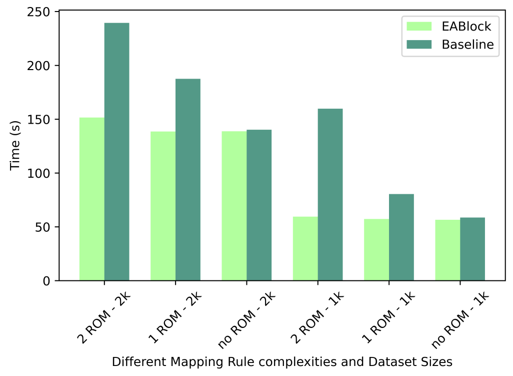
13.1 *EABlock* Efficiency- RQ1

To evaluate how the performance of a KG creation pipeline may be impacted applying EABlock, we set up 24 KG creation pipelines in overall. Experiments are

[38]`https://labs.tib.eu/sdm/biofalcon/`



(a) The performance of a KG creation pipeline applying RocketRML.



(b) The performance of a KG creation pipeline applying SDM-RDFizer.

Figure 13.1: **Efficiency**. The impact of using *EABlock* in KG creation pipelines applying two different RML-compliant engines. Baseline corresponds to the execution of entity alignment in a pre-processing stage, while *EABlock* enables the specification of this process in the RML mapping rules. As observed, *EABlock* reduces the execution time of KG creation pipelines that involve entity alignment tasks in comparison to the application of the same functions but during a pre-processing stage.

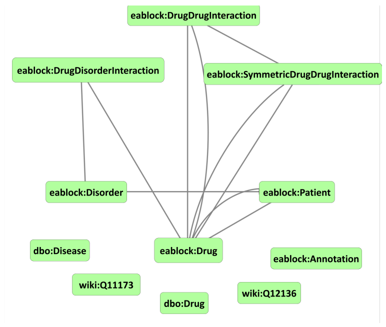
grouped as Baseline or *EABlock*; Baseline corresponds to the pipelines where execution of EA is in a pre-processing stage, while *EABlock* represent the KG creation pipelines in which *EABlock* enables the specification of EA in the RML mapping rules. Experiments are grouped into six categories, each category utilizing a different DE, i.e., all the experiments in one category have the same DE. To avoid any bias caused by the techniques applied in the development of the state-of-the-art engines, we repeat the same experiments by two different available engines including RocketRML [39] and SDM-RDFizer [40]. Accordingly, the experiments in one category differs in **a.** the applied RML-compliant engine and **b.** whether *EABlock* is used as part of the pipeline or not. **Testbeds.** Considering the parameters that affect the performance of KG creation pipelines [12], we define three different sets of mapping rules, which are distinguished based on the complexity of the `rr:TriplesMap`s that refers to the *EABlock* transformation functions. We manipulate the complexity of the mentioned rules by having different number of `rr:RefObjectMap`s, i.e., zero, one, or two `rr:RefObjectMap` (referred to as noROM, 1ROM, and 2ROM respectively, in Figure 13.1). In an attempt to prevent possible effects of data volume on the results of the experiments, we generate two relatively small datasets including 1,000 and 2,000 randomly selected records. Each dataset comprises 22 attributes, two of which are referenced in the mapping rules.

Setups. We define two KG creation pipelines, *Baseline* and *EABlock*, which execute the same entity alignment tasks and produce the same KG. The *Baseline* pipeline evaluates RML mapping rules while the entity alignment is performed in a pre-processing step. Contrary, the *EABlock* pipeline encapsulates these tasks in the *EABlock* functions that are called in the RML mapping rules. **Metrics.** *Execution time:* Elapsed time spent by the whole pipeline to complete the creation of a KG; it is measured as the absolute wall-clock system time as reported by the `time` command of the Linux operating system. The experiments were run in an Intel(R) Xeon(R) equipped with a CPU E5-2603 v3 @ 1.60GHz 20 cores, 64GB memory and with the O.S. Ubuntu 16.04LTS.

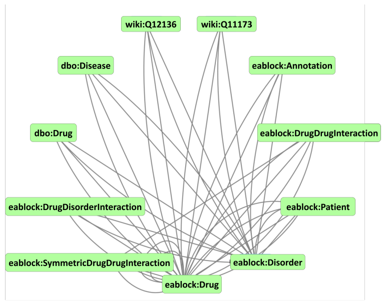
Results. Figure 13.1 illustrates the performance of two approaches of KG creations i.e., Baseline which perform the EA as pre-processing, and *EABlock* which enables the specification of EA as part of the RML mapping rules. As it can be observed in Figure 13.1, independent of the applied RML-compliant engine utilizing *EABlock* in all KG pipelines reduces the overall execution time of the KG creation. Figure 13.1

[39] `https://github.com/semantifyit/RocketRML`
[40] `https://github.com/SDM-TIB/SDM-RDFizer`

demonstrates that performing EA as pre-processing is more expensive than using *EABlock* as part of the main pipeline of KG creations. It can also be observed that in case of having more complex mapping rules, the impact of *EABlock* in decreasing the execution time is even more considerable and significant.



(a) Directed Labelled Graph G_b



(b) Directed Labelled Graph $G_{eablock}$

Analysis	**EABlock** ($G_{eablock}$)	**Baseline** (G_b)
Number of nodes	11	11
Number of edges	39	10
Avg. number of neighbors	3.091	1.273
Network diameter	2	1
Clustering coefficient	0.184	0.098
Network density	0.155	0.064
Connected components	1	6

(c) Graph metrics for G_b and $G_{eablock}$

Figure 13.2: **Connectivity Analysis**. Baseline and *EABlock* pipelines generate KG_b and $KG_{eablock}$, respectively. KG_b and $KG_{eablock}$ have the same classes and entities. However, KG_b does not include the entity alignments to UMLS, Wikidata, and DBpedia added to $KG_{eablock}$. G_b and $G_{eablock}$ are directed labelled graphs that provide an aggregated representation of KG_b and $KG_{eablock}$. The values of the graph metrics corroborate that connectivity is increased by entity alignment performed by the *EABlock* pipeline.

13.2 *EABlock* Effectiveness - RQ2

We define two pipelines: *Baseline* and *EABlock*; the *Baseline* pipeline includes no entity alignment task. The aim is to evaluate the connectivity in a KG created using the *EABlock* pipeline and assess RQ2. **Testbeds.** For this group of experiments we utilize the TestBed 4, described in Chapter 10. **Analysis.** Let KG_b and $KG_{eablock}$ be the KGs created by the *Baseline* and *EABlock* pipelines, respectively. $KG_{eablock}$ comprises 10,339,870 RDF triples, while KG_b has 10,200,209. KG_b and $KG_{eablock}$ are used to create two labelled directed graphs $G_b = (V, E_b)$ and $G_{eablock} = (V, E_{eablock})$ and traditional network analysis methods are applied to determine connectivity. Vertices in V keeps the classes in KG_b and $KG_{eablock}$ with at least one resource; KG_b and $KG_{eablock}$ have the same resources and literals. A labelled directed edge $e = (q, p, k)$ belongs to E_b (resp. to $E_{eablock}$) if there are classes Q and K in V, and q and k are instances of Q and K in KG_b (resp., $KG_{eablock}$) and the RDF triple (qpk) belongs to KG_b (resp., $KG_{eablock}$). G_b and $G_{eablock}$ provide an aggregated representation of KG_b and $KG_{eablock}$. Figure 13.2 depicts G_b and $G_{eablock}$; $G_{eablock}$ is composed of 11 vertices and 39 directed edges. While, G_b comprises 11 vertices connected by only 10 edges. Table 13.2c compares G_b and $G_{eablock}$ in terms of graph metrics generated by Cytoscape[41]. **Metrics. a.** Average number of neighbors indicates the average connectivity of a vertex or node in a graph. **b.** Network diameter measures the shortest path that connects the two most distant nodes in a graph. **c.** Clustering coefficient measures the tendency of nodes who share the same connections in a graph to become connected. If a neighborhood is fully connected, the clustering coefficient is 1.0 while a value close to 0.0 means that there is no connection in the neighborhood. **d.** Network density measures the portion of potential edges in a graph that are actually edges; a value close to 1.0 indicates that the graph is fully connected. **e.** The number of connected components indicates the number of subgraphs composed of vertices connected by at least one path. **Results.** The results reported in Figure 13.2c indicate the average number of neighbors in $KG_{eablock}$ comprises entities that are more connected. Moreover, the clustering coefficient is relatively low, but the CUIs annotations and links to DBpedia and Wikidata included in $KG_{eablock}$, increase the connectivity in the neighborhoods of $G_{eablock}$. In particular, `eablock:Patient`, `eablock:DrugDisorder`, `eablock:Annotation`, `eablock:Disea- se`, `wiki:Q12136`, `wiki:Q11173`, and `dbo:Drug` have a neighborhood connectivity of 8 in $G_{eablock}$. On the other hand, in G_b, the neighborhood connectivity of `eablock:Annotation`, `eablock:Di- sease`, `wiki:Q12136`, `wiki:Q11173`, and `dbo:Drug` is 0, and `eabloc- k:Patient` and `eablock:DrugDisorderInteraction` is 3. These results corroborate that connectivity is enhanced as a result of the entity alignment implemented by the *EABlock* functions.

13.3 Summary

In this chapter, we empirically evaluated *EABlock*, an approach to solve entity alignment problems presented in Chapter 7. We observed that with an eager evaluation strategy and efficient translation of mapping rules into function-free rules, applying *EABlock* ensures not to sacrifice efficiency at the cost of reproducibility. The observed experimental results show the benefits of grounding solutions for KG creation in well-established problems like NER, NL, and data integration systems.

[41]`https://cytoscape.org/`

Chapter 14

Evaluating Dragoman

In Chapter 9, we propose a solution to the RQ3. In this chapter, we evaluate the performance of our proposed solution. We empirically prove that the proposed techniques ensure the efficiency of materialized data integration systems. We report two groups of experiments including *preliminary* and the *extended* empirical studies. To the best of our knowledge, there is no experimental study focusing on the performance of declarative knowledge graph creation frameworks that include user-defined functions. That leads us to perform the *preliminary* experimental study in order to gain some insight into the specific parameters that may impact such pipelines. Considering the observations made during the *preliminary* experiments, we set up the *extended* empirical study as the extended version involving more influential parameters or suspected influential ones. The experiments described in this chapter are partially reported in [41] and under review in our very latest paper [42].

14.1 *Preliminary* Experimental Study

The aim of this preliminary experimental study is to answer the following questions considering the first in-progress version of *Dragoman*, named *FunMap*[43].

RQ1) What is the impact of data duplication rate on the execution time of a knowledge graph creation approach?
RQ2) What is the impact of different types of complexity over transformation functions during a knowledge graph creation process?

[42]`https://www.semantic-web-journal.net/content/dragoman-efficiently-evaluating-declarative-mapping-languages-over-framework`
[43]`https://doi.org/10.5281/zenodo.3993657`

RQ3) How does the repetition of the same user-defined function in different mapping assertions affect the existing RML engines?
RQ4) What is the impact of relational data sources in the knowledge graph creation process?

All the resources used to perform this evaluation are available in our GitHub repository[44]. The testbeds that we apply in the *preliminary* study are TestBed 2, Group i which are explained in detail in Subsection 10.4.2. In addition to these testbeds, we also validate *FunMap* in the case of large-sized data, with a dataset following of random collection of 4,000,000 records and a size of about 1.3GB.

Engines. The baselines of our study are three different open source RML-compliant engines that are able to execute RML+FnO mappings and have been extensively utilized in multiple applications and tested by the community: SDM-RDFizer v3.0 [36], RMLMapper[45] v4.7, and RocketRML[46] v1.1. [47]. In order to evaluate the impact of transformation rules, we implement the first version of FunMap on top of the aforementioned engines with source-based projection as an optional parameter. We refer to the approach which applies FunMap excluding source-based projection as FunMap^{-}[48]. We created a docker image per tested engine for reproducibility[49].

Metrics. *Execution time:* Elapsed time spent by an engine to complete the creation of a knowledge graph and also counts FunMap pre-processing; it is measured as the absolute wall-clock system time as reported by the `time` command of the Linux operating system. Each experiment was executed five times and average is reported. The experiments were executed on an Ubuntu 16.04 machine with Intel(R) Xeon(R) Platinum 8160, CPU 2.10GHz and 700Gb RAM.

[44] `https://github.com/SDM-TIB/FunMap`

[45] `https://github.com/RMLio/rmlmapper-java`

[46] `https://github.com/semantifyit/RocketRML/`

[47] We name them SDM-RDFizer**(RML+FnO), RMLMapper**(RML+FnO), and RocketRML**(RML+FnO).

[48] We name these combined engines as follows: a) FunMap: FunMap+SDM-RDFizer, FunMap+RMLMapper, and FunMap+RocketRML; b) FunMap^{-}: FunMap^{-}+SDM-RDFizer, FunMap^{-}+RMLMapper, and FunMap^{-}+RocketRML.

[49] Our paper has been accepted as the "fully-reproduced paper" in ISWC 2020 conference.

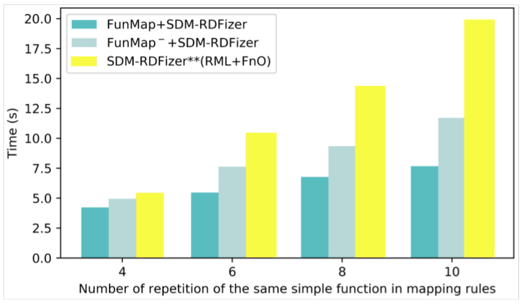
14.1.1 Experimental setups.

Based on our research questions, we set up overall 198 experiments as the combinations of the following scenarios. We create two datasets from our baseline with 25% and 75% duplicates which means in the 25% duplicate dataset, 25% and in the 75% duplicate dataset, 75% of the records are duplicated. Additionally, two functions with different levels of complexity are created. We describe the complexity level of the functions based on the number of required input attributes and operations to be performed. Accordingly, "simple" function is defined to receive one input attribute and perform one operation, while a "complex" function receives two input attributes and completes five operations. In total, we create eight mapping files including four, six, eight, and ten `TriplesMap` and one `FunctionMap` to be either "simple" or "complex". Additionally, six experiments using 75% duplicate datasets of 20,000 and 4,000,000 records and a mapping file including ten complex functions are set up in order to be run over a relational database (RDB) implemented in MySQL 8.0 [50].

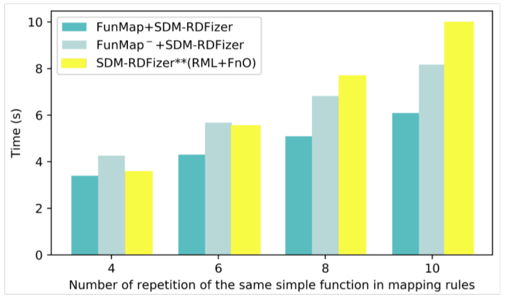
14.1.2 Observations and Results

In this section, we describe the outcomes of the *preliminary* experimental evaluations. Figure 14.1 reports on the execution time of the different testbeds in which the functions are considered to be "simple" whereas Figure 14.2 shows the experiments involving "complex" functions. Both figures represent the total execution time for constructing the knowledge graph applying selected engines (i.e., SDM-RDFizer, RMLMapper, and RocketRML) in three different configurations: a) the current version of the engine that is able to directly interpret RML+FnO mappings in the engine (e.g., RMLMapper**(RML+FnO)); b) FunMap^{-} in conjunction with the engine (e.g., FunMap^{-}+RMLMapper); and c) FunMap together with the engine (e.g., FunMap+RMLMapper). In the case of all the configurations of RocketRML, we only provide the results for the execution of simple functions because the engine does not execute joins with multiple conditions[51] correctly, hence, the proposed optimizations cannot be applied. For the rest of the experiments, we have verified that the results are the same for all the approaches in terms of cardinality and correctness.
The results obtained by the application of SDM-RDFizer with the repetition of simple functions (Figures 14.1a and 14.1b) reflect an improvement of the execution time when FunMap is applied in the process. With the growth of the number of duplicates and repeated functions, the difference between the performance of SDM-

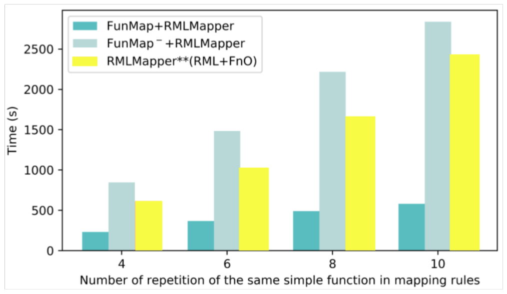
[50]https://www.mysql.com/
[51]Check an example in the zip file of the supplementary material.



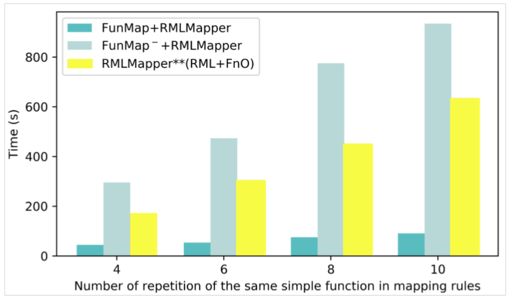
(a) SDM-RDFizer - 25% of duplicates



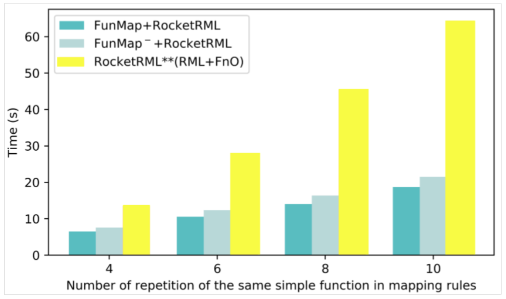
(b) SDM-RDFizer - 75% of duplicates



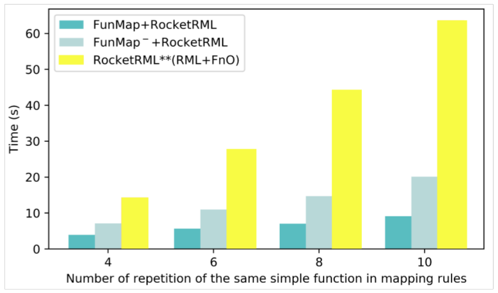
(c) RMLMapper - 25% of duplicates



(d) RMLMapper - 75% of duplicates



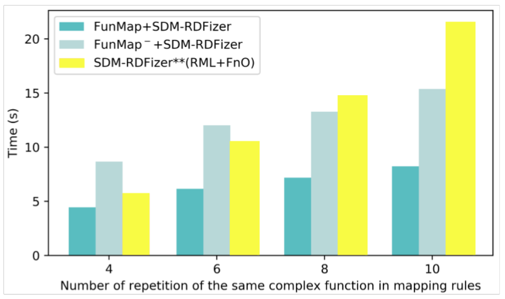
(e) RocketRML - 25% of duplicates



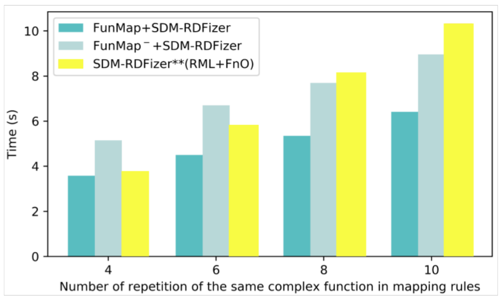
(f) RocketRML - 75% of duplicates

Figure 14.1: **Total execution time of experiments with simple functions 25-75% of duplicates.** SDM-RDFizer, RMLMapper and RocketRML executing simple functions in RML+FnO mappings and with FunMap and FunMap$^-$.

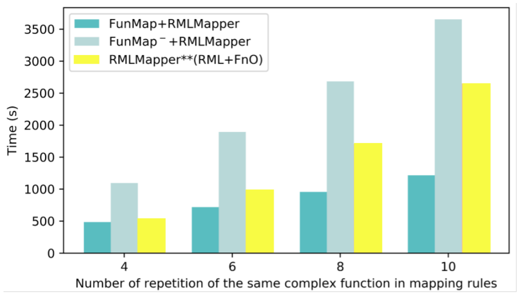
RDFizer**(RML+FnO) and FunMap+SDM-RDFizer increases. Using this engine, FunMap$^-$ shows the same behavior as FunMap, however, in the case of having a large number of duplicates and a few repeated functions FunMap$^-$ does not improve the performance of SDM-RDFizer**(RML+FnO). In the case of using RMLMapper (Figures 14.1c and 14.1d), we observe that the results obtained together with



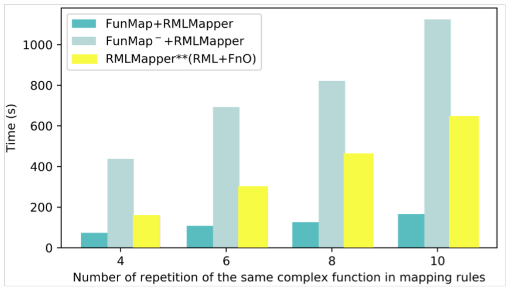
(a) SDM-RDFizer - 25% of duplicates



(b) SDM-RDFizer - 75% of duplicates



(c) RMLMapper - 25% of duplicates



(d) RMLMapper - 75% of duplicates

Figure 14.2: **Total execution time for complex functions 25-75% of duplicates.** SDM-RDFizer and RMLMapper executing complex functions in RML+FnO mappings and with FunMap and FunMap$^-$.

FunMap$^-$ (i.e., DTR1 optimization) do not show better performance than RMLMapper**(RML+FnO). DTR1 which only focuses on transforming functions, delegates the removal of the duplicates to the engine which is not accomplished efficiently by RMLMapper. However, in FunMap+RMLMapper, which includes DTR1 and DTR2 optimizations, duplicates are removed before the execution of the RML mappings and leading to obtaining results that clearly show improvements with respect to the baseline. In the same manner, as the SDM-RDFizer, the number of repetitions of the functions affects the execution time of the RMLMapper**(RML+FnO), while FunMap maintains similar execution times. Finally, RocketRML (Figures 14.1e and 14.1f) seems not to be affected by the number of duplicates over the input data, obtaining similar execution times for 25% and 75% rate for RocketRML**(RML+FnO). However, the number of repetitions over functions impacts the performance of RocketRML**(RML+FnO), increasing the total execution time. The incorporation of DTR1 (i.e., FunMap$^-$+RocketRML) and DTR2 (i.e., FunMap + RocketRML) enhances the performance and scalability during the construction of the knowledge graph, obtaining a similar behavior as the other two tested engines.

The effect of function complexity over SDM-RDFizer can be observed in Figures 14.2a and 14.2b. Whenever the number of repetitions is low (4-6), the join with multiple conditions affects FunMap$^-$ + SDM-RDFizer, obtaining worse results than SDM-RDFizer**(RML+FnO). However, if repetitions increase (8-10), DTR1 empowers SDM-RDFizer**(RML+FnO) due the reduction of repeated operations during the evaluation of the mappings. Conversely, FunMap+SDM-RDFizer exhibits better results than SDM-RDFizer**(RML+FnO) in all the testbeds. Finally, the behavior of RMLMapper – when it has to execute complex transformation functions (Figures 14.2c and 14.2d)– is affected in terms of execution time for the configuration FunMap$^-$+RMLMapper in comparison to the case of simple functions. As similar as SDM-RDFizer, the join with several conditions is impacting the performance. However, together with data transformation optimizations, FunMap+RMLMapper outperforms the baseline.

The experimental results on RDBs show even more significant improvement in the performance of both RMLMapper and SDM-RDFizer in the presence of FunMap. In FunMap+RMLMapper, applying `join`s in the SQL queries that define the `logicalSource`s instead of using `joinCondition`s reduces execution time by up to a factor of 18. These results evidence that `joinCondition`s are not efficiently implemented by RMLMapper, and explain why FunMap+RMLMapper is showing less improvement compared to FunMap+SDM-RDFizer in Figure 14.2. Moreover, FunMap+SDM-RDFizer successfully performs on the large-sized relational dataset of 1.3GB in 5,670.67 seconds, while SDM-RDFizer**(RML+FnO) cannot create the KG and times out after 10,000 seconds.

In overall, we observe that the configurations that interpret RML+FnO mappings directly are affected by the repetition of the functions and the degree of data duplicates, i.e., execution time monotonically increases with the number of functions and data duplication degree. In contrast, the incorporation of FunMap into the engines shows less fluctuated behavior when the data duplication rate increases. Additionally, the studied engines handle the repetition of the functions during the construction of the knowledge graph thanks to the pushing down of the execution of the functions directly over the dataset. In summary, the observed results indicate that the FunMap heuristics improve the performance of data integration systems and generate solutions to the problem of scaled-up knowledge graph construction. The effectiveness of the proposed transformations has been empirically demonstrated on various

RML+FnO and RML-compliant engines. However, we observe that there are cases where the application of DTR1 alone is not enough (i.e., FunMap$^-$), being required the applications of all the transformations (i.e., DTRs and MTRs) to provide an effective solution.

14.2 *Extended* Experimental Study

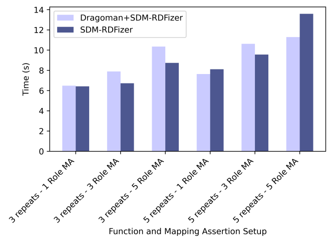
Observing the results of the *preliminary* experiments, we set up the extended experimental study, i.e., the *extended* study to evaluate Dragoman. To assess the performance of Dragoman we aim to answer the following questions:

RQ1) What is the impact of applying Dragoman in a knowledge graph creation pipeline in terms of execution time?
RQ2) What parameters illustrate the advantages of applying Dragoman in knowledge graph creation pipelines?

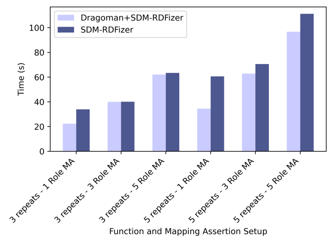
To answer the research questions, we set up 220 different pipelines of knowledge graph creation, half of which deploy Dragoman to perform the tasks of user-defined functions evaluation and data integration system transformation, following an eager evaluation. While the rest of the pipelines rely on the same engines which generate the RDF triples to also perform the task of function evaluations, based on a lazy evaluation strategy. With the mentioned comparison, we aim to observe whether the execution time required to generate the same knowledge graph reduces once Dragoman is applied in the pipeline.

14.2.1 Implementation

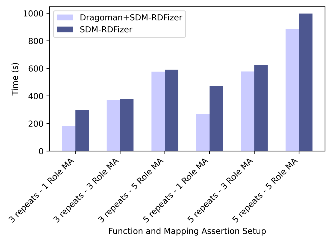
Dragoman is implemented in Python3. As a proof of concept, the implementation includes the transformation of two data source formats: CSV files and relational databases (RDB). It should be noted that there are small differences in implementing the transformation rules between CSV and RDB. As explained previously, the data source resulting from the concept-based transformation is the join between two data sources. In the case of having data sources as CSV files, Dragoman stores the materialization of the joins between the sources in CSV files. However, in the case of having the data sources as relational tables, instead of materializing the joins, we add the SQL join queries in the mapping assertions enabling real-time materialization. Dragoman is open-source and licensed under Apache License 2.0. It is publicly



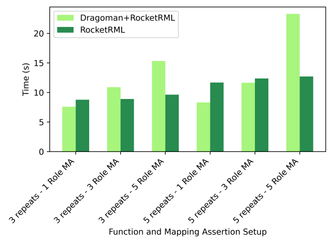
(a) 10k records - NonInjSurj



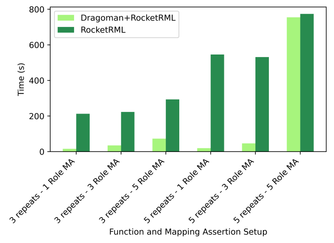
(b) 100k records - NonInjSurj



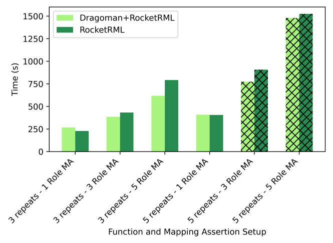
(c) 1M records - NonInjSurj



(d) 10k records - NonInjSurj



(e) 100k records - NonInjSurj



(f) 1M records - NonInjSurj

Figure 14.3:

accessible through a GitHub repository[52] and Zenodo[53].

Functions. As we explained earlier, the same experiments are repeated twice to compare the execution time of different knowledge graph creation pipelines in the presence and absence of Dragoman. In other words, we create the same knowledge graph, once using Dragoman to perform the evaluation of the functions and then another engine to generate the RDF triples, and the second time, only using an engine to perform both tasks of the function evaluation and RDF triples generation. In that regard, we can only use RML-compliant engines that are capable of executing user-defined functions. We also consider the empirical study conducted in [36] and [41] to only select the engines that are efficient in executing multi-sources role mapping assertions, i.e., SDM-RDFizer[54] and RocketRML[55] [56]. Furthermore, the

[52]`https://github.com/SDM-TIB/Dragoman`

[53]`https://doi.org/10.5281/zenodo.6418124`

[54]`https://github.com/SDM-TIB/SDM-RDFizer`

[55]`https://github.com/semantifyit/RocketRML`

[56]RMLMapper is another engine capable of executing functions, while, it is inefficient executing multi-sources role mapping assertions according to the experimental results reported in [36, 41]

implementation of test functions is required to be added to the chosen engines in order to equip them with the evaluation of these functions. Following the languages in which these engines are developed, we implement our test functions in SDM-RDFizer using Python and in RocketRML using Javascript.

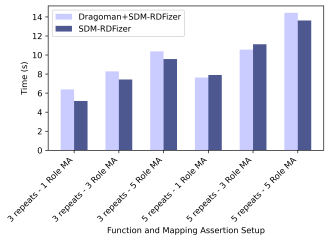
Table 14.1

Studied Parameters			Experiments Groups		
			Efficiency	Complex Joins	Composite Functions
Data Source	Par1: Data Size	10K	✗	✗	✗
		100K	✗	✗	✗
		1M	✗	✗	✗
	Par2: Selectivity	Low		✗	✗
		Medium		✗	✗
		High		✗	✗
Mapping Assertion	Par3: # of appearances of the same user-defined functions	2		✗	
		3	✗		
		4		✗	
		5	✗		
	Par4: Type of mapping assertions that include user-defined functions	Concept Mapping Assertion		✗	✗
		Role Mapping Assertion	✗	✗	
	Par5: The overall # of role mapping assertions	1	✗		
		3	✗		
		5	✗		
	Par6: Star Joins	Star joins shaped as the result of Dragoman (DIS')		✗	✗
		Star joins exist in the original DIS		✗	
	Par7: Chain Joins	1		✗	
Function	Par8: Types of user-defined functions	Non-Injective Surjective	✗		
		Bijective	✗		
	Par9: Composite user-defined functions	a(b(c(.)))			✗

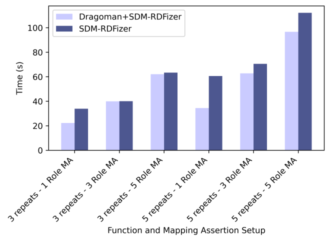
14.2.2 Experimental Setups

Considering the number of parameters required to be studied, the number of control variables for each experiment is also significant. To better observe and understand, we categorize the experiments into three groups according to the three most challenging parameters that have not been considered in the similar studies [41] i.e., function type, composite function, and complex multi-sources role mapping assertions.

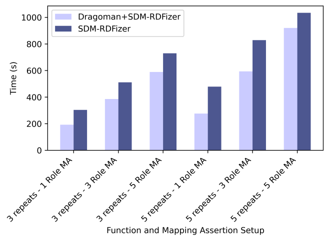
Setups. To answer RQ1, we define the process of knowledge graph creation as the combination of two tasks, including the execution of the functions and the translation of the mapping rules into RDF triples. Therefore, we execute the knowledge graph creation pipeline twice for each experiment setup. The first execution considers the same RML-compliant engine to perform both the tasks of the function execution and RDF triples generation. However, in the second attempt, the tasks of executing the functions and providing function-free mapping assertions are assigned to Dragoman, while the generation of the RDF triples is performed by an RML-compliant engine



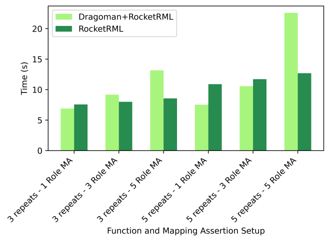
(a) 10k records - Bijective



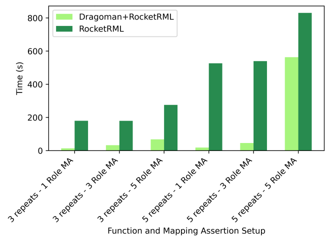
(b) 100k records - Bijective



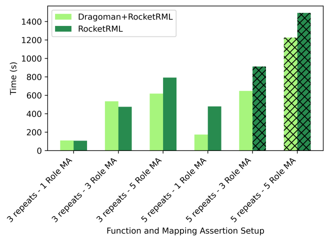
(c) 1M records - Bijective



(d) 10k records - Bijective



(e) 100k records - Bijective



(f) 1M records - Bijective

Figure 14.4:

according to the transferred DIS provided by Dragoman.

Metrics. *Execution time:* Elapsed time spent by the whole pipeline, including function execution to complete the creation of a Knowledge graph; it is measured as the absolute wall-clock system time as reported by the `time` command of the Linux operating system. The timeout is set to be five hours. The experiments were run in an Intel(R) Xeon(R) equipped with a CPU E5-2603 v3 @ 1.60GHz 20 cores, 64 G.B. memory, and the O.S. Ubuntu 16.04LTS.

Datasets and Mapping Assertions. For these experiments, we rely on Testbed 2, Group ii.

14.2.3 Efficiency

We set up 144 experiments in this category to study four different parameters, including Par1, Par3, Par5, and Par8. To avoid any possible bias generated by the engines translating mapping assertion engines, we evaluate the same experiments in this category with two recognized RML-compliant engines that support functions execution, i.e., RocketRML v1.12.0[57] and SDM-RDFizer v4.0[58]. In other words, we set up 72 different DIS and observed the creation of knowledge graphs from each DIS twice, using one of the two engines. It should be noted that in choosing the RML-compliant engines that support function execution, our criteria is the performance of the engines translating multi-sources role mapping assertions [36]; the two selected engines have competitive performance translating multi-sources role mapping assertions.

Observations and Results. The results of this group of experiments using the non-injective surjective function are visualized in Figure 14.3 while the results of the same experiments using the bijective function type are summarized in Figure 14.4. Figure 14.3a, Figure 14.3b, Figure 14.3c, Figure 14.4a, Figure 14.4b, and Figure 14.4c illustrate - in purple - the results of executing the set-up knowledge graph pipelines using SDM-RDFizer as the RML-compliant, while, the other six subfigures including Figure 14.3d, Figure 14.3e, Figure 14.3f, Figure 14.4d, Figure 14.4e, and Figure 14.4f present - in green - the results of executing the same pipelines applying RocketRML as the RML-compliant engine. The lighter colored bars in all the sub-figures Figure 14.3 and Figure 14.4 show the results utilizing Dragoman for executing the functions and transformation of the pipelines and applying the RML-compliant engine for translating the output of Dragoman into the RDF knowledge graph. In contrast, the darker colored bars in Figure 14.3 and Figure 14.4 represent the results of executing the whole knowledge graph creation pipeline, i.e., including the functions evaluations, using the RML-compliant engines. The textured bars present the results of the experiments in which the engine is unable to generate the complete result.

Observations on Par1: Comparing the execution time required by the pipelines, including Dragoman, with the same ones without applying Dragoman clearly shows the advantages of utilizing Dragoman in the case of large-size data sources. In the case of having small data sources, i.e., 10k, we observe no cost savings. This means that the transformations performed by Dragoman add more overheads rather than reducing the cost. However, in the case of 100k and 1M, we can observe significant benefits in applying Dragoman. It should be noted that the results of the same experiments differ when different RML-compliant engines are applied. This can be due to the different algorithms and performances that engines have, which is out of the scope of this empirical study.

[57]`https://github.com/semantifyit/RocketRML`
[58]`https://github.com/SDM-TIB/SDM-RDFizer`

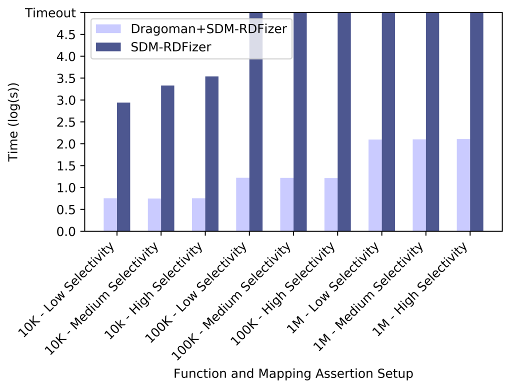
Observations on Par3 and Par5: Comparing the performances of the pipelines with increasing numbers of the appearances of the same functions or numbers of role mapping assertions suggests that applying Dragoman can improve the performance significantly while the numbers increase. Nevertheless, applying Dragoman in the pipelines with small numbers of Par3 or Par5 can increase the required execution time.

Observations of Par8: In overall, we can observe in Figure 14.3 and Figure 14.4 that the results of the same experiments, i.e., the pipelines with the same values for the Par1, Par3, and Par5, show the same patterns using different values for Par8. In other words, these results do not support our hypothesis about the impact of the type of functions on knowledge graph creation.

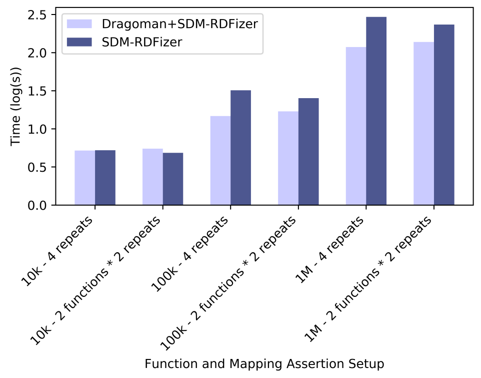
Complex Joins

The main focus of this category of experiments is to study the impact of Par6 and Par7 in different experimental scenarios. Accordingly, these experiments can be divided into two sub-categories; Chain Join studies the impact of Par7, and Star Join focuses on Par6. Both categories study the impact of Par1 considering three different data sizes, i.e., 10k, 100k, and 1 million records. However, they both consider only one value for Par3, which is four appearances of the same user-defined function. *Chain Join.* This group of experiments involves one chain join as the value of par7, however, they adopt different values of Par2, i.e., low or 80%, medium or 50%, and finally, high or 20% join selectivity. This setup aims to study the impact of chain join in the presence of different selectivity rates. *Star Join.* In these experiments, we focus on the Par6 parameter, a.k.a star join. Since star join is an expensive logical operation for an engine, it is very important to be studied in the empirical evaluation of any knowledge graph creation engine. According to the experimental study by Iglesias et al. [37], SDM-RDFizer can perform star joins, which RocketRML fails to do. Therefore, we utilize SDM-RDFizer in this group of experiments.

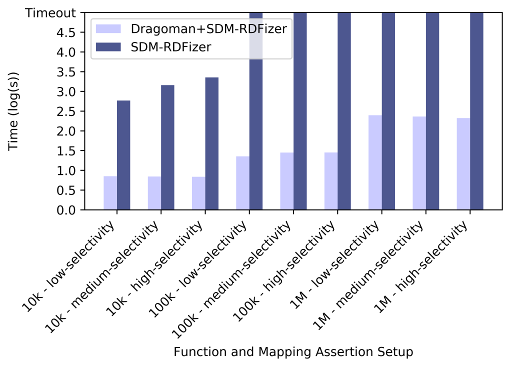
In overall, we set up nine knowledge graph creation pipelines composed of join stars; we aim to observe the cost of executing functions using Dragoman in such expensive pipelines. In these testbeds, three different values of Par2, i.e., low, medium, and high join selectivity are considered. Note that in these nine testbeds, functions are not repeated. Furthermore, in this group of experiments, we also study Par4 by including the user-defined functions in the concept mapping assertion of the first



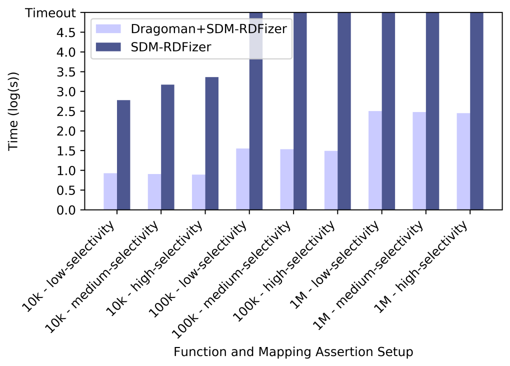
(a) Join Chain



(b) Star Join in Dragoman result



(c) Star Join in Original DIS - Simple Function



(d) Star Join in Original DIS - Composite Function

Figure 14.5:

nine experiments, contrary to the previous experiments in which the same functions appear as role mapping assertions.

Observations and Result. Figure 14.5a summarizes the results of the experiments in the chain join category, while, Figure 14.5c illustrates the results of the experiments in the star join category. Considering the wide range of execution time values of these experiments, we visualize the results' logarithm value for clarity.

Observations on Chain Join; Par1, Par2, and Par7: As it can be observed in Figure 14.5a, applying Dragoman significantly improves all knowledge graph creation pipelines' performance. Contrary to the previous group of experiments, i.e., Subsection 14.2.3, these experiments, including chain join, show improvement independent of the size of the data sources. Another important observation to note is the impact of join selectivity. Although the quantity of savings in experiments with different join selectivity differs, the reduced time is considered in all the cases.

Observations on Star Join, Par1, Par2, Par4 and Par6: Similar to the results obtained from chain join experiments, as it can be observed in Figure 14.5c, utilizing Dragoman improves the performance of knowledge graph creation pipelines in all the nine cases which include star joins in their original mapping assertions. In other words, the only difference in the observed results of the setups with various values for Par1 or Par2 is the quantity of the savings; the higher the data size or selectivity, the greater the savings.

Composite Functions

Testbeds. In this group of experiments, we aim to study the impact of composite user-defined functions in mapping assertions, i.e., Par9. To this end, we repeat the same nine experiments of the first category of star join explained earlier. We repeat these nine experiments explained in Section 14.2.3 with composite functions in the form of $A(B(C(.)))$.

Observation and Results. In addition to the results of the experiments with simple functions illustrated in Figure 14.5c, the results of the same experiments with composite functions can be observed in Figure 14.5d. Comparing Figure 14.5c and Figure 14.5d, we can conclude that the complexity of a function in terms of being composite has no significant impact on the overall execution time of the knowledge graph creation pipeline.

14.2.4 Summarizing the Results

RQ1. The answer to this research question is studied in all three categories of the experiments by dividing the knowledge graph creation process into two tasks. The impact of performing the task of functions evaluation and transforming the DIS into a function-free one by Dragoman is compared with the performance of two other RML-compliant engines for the same tasks. The common observation among most of the experiments shows that in the case of having a small data source, i.e., 10k, the optimization performed by Dragoman costs more than the amount of saving that it presents to the process of knowledge graph creation. In contrast, applying Dragoman

in creating knowledge graphs from larger data sources, i.e., 100k and 1M, optimizes the overall cost. The “complex multi-sources role mapping assertions” category of the experiments is exempted from explained conclusions; they show savings even with small datasets.
RQ2. Data size. As expected, based on similar studies, the data source size is an important factor in deciding the application of Dragoman. It should not be interpreted that the size of the data source impacts the performance of the Dragoman, but rather it impacts the number of optimization benefits that are brought by Dragoman. **Join selectivity.** As expected, the results of the “complex multi-sources role mapping assertions” illustrate that join selectivity is an effective parameter in knowledge graph creation pipelines. In contrast, the results of the same experiments which utilize Dragoman show no significant difference in the execution time between data sources with various join selectivity. **Mapping Assertions.** As expected, due to eager evaluation, Dragoman improves the execution time of knowledge graph creation processes that involve repeating the same user-defined function in different mapping assertions. In contrast, no specific impact on the overall performance of Dragoman can be observed considering the parameter Par4, i.e., the type mapping assertion that involves user-defined functions. The effect of the parameters Par6 and Par7 are studied by the “complex multi-sources role mapping assertions” group of experiments. The significant improvements observed in using Dragoman in these results position Dragoman as a requirement in any knowledge graph creation pipeline, including functions and complex joins. Because even simple joins can be expensive, providing optimization for executing complex joins and transforming the DIS, which times out into an efficient one, show obvious use cases of Dragoman. **Function Parameters.** It can be observed from the results of the first category of the experiments, i.e., “function type”, that the type of the function has no obvious impact on the overall execution time of knowledge graph creation pipelines. The result of the “star join” experiments reveal the same conclusion for the “function composition” parameter. Nonetheless, it should be noted that it also implies that the “function composition” parameter has no impact on the performance of Dragoman.

14.3 Summary

In this chapter, we conduct experimental studies to investigate if utilizing Dragoman to execute the user-defined functions and transforming the DIS into a function-free one reduces the required execution time. As observed in the results, the application of Dragoman in knowledge graph creation pipelines from large data sources can reduce overall costs significantly; up to 75% savings. It is also discovered that with complex mapping assertions such as “star join”, utilizing Dragoman always decreases knowledge graph creation costs, albeit with small data sources.

Part IV

Applications and Conclusion

Chapter 15

Applications

In this chapter we focus on answering the following question:

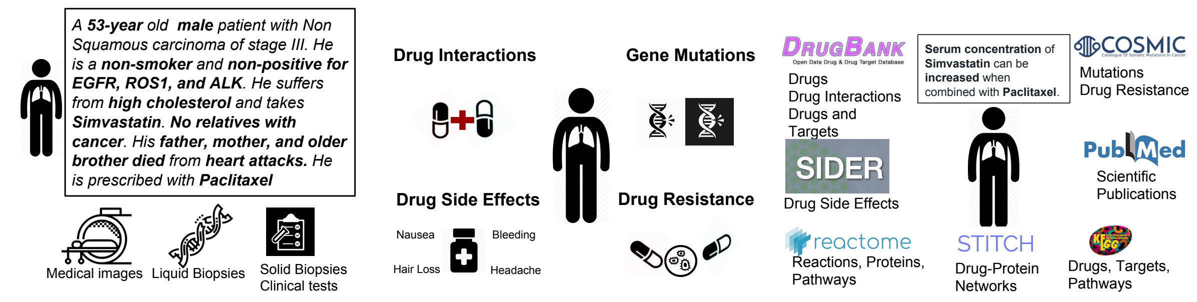
RQ5: How to overcome the challenges in applying materialized data integration systems in real-world scenarios?

To contribute to the mentioned research question, we explain how we apply our proposed techniques in four different real-world projects overcoming the existing challenges. We show why and how the optimization techniques proposed in this thesis were required for succeeding in these projects with respect to the objectives and the KPIs. As a proof of concept, we provide one generic use case in the context of lung cancer pilots in EU H2020 funded personalized medicine project iASiS[59], BigMedilytics[60], and CLARIFY [61]. The content of this chapter is based on our collaborative research that is published in [70, 73, 71, 56, 72, 73].

15.1 Motivating Example

Personalized medicine including personalized diagnosis, prognosis, and treatment demands gaining broad insight into patients' characteristics. For this purpose, patients' data which can be scattered over a wide variety of sources is required to be integrated and analyzed. More specifically, the analysis of the services visited the most by a patient before a new diagnosis and the type of requested tests may uncover patterns

[59]`https://project-iasis.eu/`
[60]`https://www.bigmedilytics.eu/`
[61]`https://www.clarify2020.eu/`


*A **53-year** old **male** patient with Non Squamous carcinoma of stage III. He is a **non-smoker** and **non-positive for EGFR, ROS1, and ALK**. He suffers from **high cholesterol** and takes **Simvastatin**. **No relatives with cancer**. His **father, mother, and older brother died** from **heart attacks**. He is prescribed with **Paclitaxel***

Serum concentration of **Simvastatin** can be **increased** when combined with **Paclitaxel**.

(a) Electronic Health Records (b) Impacts in Treatment Effectiveness (c) Biomedical Data Sources

Figure 15.1: **Motivating Example**. Heterogeneous sources of knowledge. (a) Unstructured data sources, e.g., clinical notes, medical images, and clinical tests, encode invaluable knowledge about a patient medical condition. (b) Factors impact on the effectiveness of a treatment; they need to be identified to increase a patient survival time. (c) Various biomedical repositories maintain knowledge collected by the scientific community about facts that can contribute to the prescription of effective treatments. Data sources range from structured (e.g, COSMIC), to unstructured (e.g., PubMed); and short texts in structured data sources may encode also relevant knowledge (e.g., drug interactions). Heterogeneity problems across sources need to be solved for extracting the required knowledge.

that contribute to earlier disease detection and treatment effectiveness.

Figure 1.1 shows an exemplary set of data sources that are received from different partners of the projects and are required to be integrated with a knowledge-driven ecosystem. Electronic health records (EHRs) Figure 1.1 preserve the knowledge about the conditions of a patient that need to be considered in order for effective diagnosis and treatment prescriptions. Albeit informative, EHRs usually preserve patient information in an unstructured way, e.g., textual notes, images, or genome sequencing. Furthermore, EHRs may include incomplete and ambiguous statements about the whole medical history of a patient. In consequence, knowledge extraction techniques are required to mine and curate relevant information for an integral analysis of a patient, e.g., age, gender, life habits, mutations, diagnostics, treatments, and familial antecedents.

In addition to evaluating information in EHRs, physicians depend on their experience or available sources of knowledge to predict potential adverse outcomes, e.g., drug interactions, side-effect or resistance (Figure 15.1b). Diverse repositories and databases make available crucial knowledge for the complete description of a patient's condition and the potential outcome (Figure 15.1c). Nevertheless, sources are autonomous and utilized diverse formats that range from unstructured scientific publications in PubMed[62] to dumps of structured data about cancer-related mutations in COSMIC [63]. To illustrate, the effect of the interactions between two drugs is reported in DrugBank like short text, e.g., the effect of the interactions between Simvastatin and Paclitaxel. In order to detect the facts that can impact the effectiveness of a particular treatment, e.g., Paclitaxel, the 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 specific treatment. All the mentioned complexities and opportunities integrating patients' data lead us to devise a data ecosystem for lung cancer patients and materialize it using the optimization techniques proposed in this thesis.

15.2 Knowledge-driven Data Ecosystems

One of the main goals in personalized medicine projects is to develop analytical tools that provide oncologists insight to improve the management of lung cancer patients during their treatment and post-treatment. Data ecosystems (DE) [54] are knowledge-driven infrastructures that allow the development of such analytical tools. DEs are furnished with various computational methods to solve interoperability and integrate data, while preserving data privacy, security, and sovereignty. The design of DEs is considered a crucial technological building block for digitalization and the digital economy of the future. DEs aim at being aligned with European Strategy Data and facilitate the creation of data markets for ensuring Europe's competitiveness and data sovereignty. Several research initiatives and industry consortia have followed DEs; they contribute with reference architectures to tackle: (i) data governance according to regulations imposed by data providers; (ii) policies and computational frameworks to ensure a trusted and secure data exchange; (iii) semantic data models for representing main data concepts and relationships, as well as exchange formats and protocols, and (iv) software design principles for guiding the implementation of the components of the reference architectures.

[62] `https://www.ncbi.nlm.nih.gov/pubmed/`
[63] `https://cancer.sanger.ac.uk/cosmic`

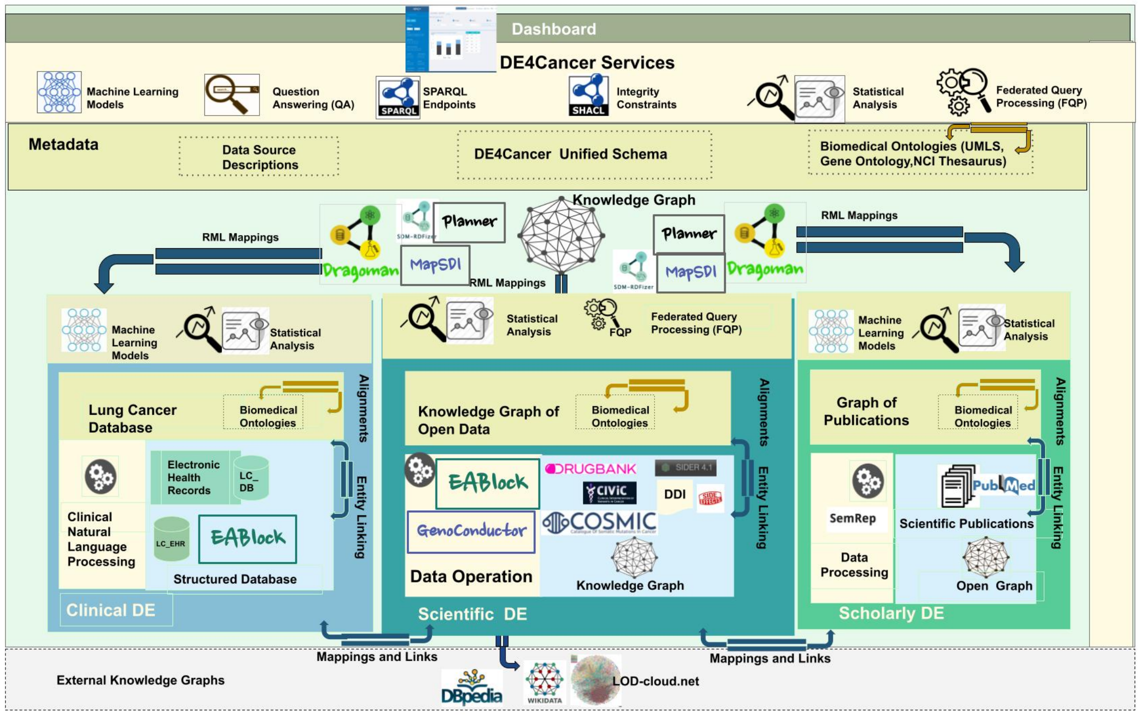


Figure 15.2: The LungCancerDE Data Ecosystem

DEs are flexible infrastructures able to fulfill requirements imposed by DE stakeholders. DEs can be centralized, and one single node maintains all the data sources shared by the providers. The node also hosts all the services implemented on top of the DE data sources. Contrary, whenever data cannot be moved to a single node and data privacy regulations hinter the materialized and complete data integration of the DE data sources, DEs will be decentralized, i.e., they will be composed of several nodes. Each DE node will be able to perform services and share data management and analytical results.

Data interoperability is a barrier in DEs; thus, semantic data models or ontologies describing the meaning of the data sources are also part of a DE. Moreover, mapping rules relating to how data sources are defined in terms of the semantic data models are included. Lastly, a DE can also be enhanced with a meta-layer that describes business models, data access regulations, and data exchange contracts.

15.3 LungCancerDE

We propose LungCancerDE, the lung cancer data ecosystem, a network of data ecosystems (DEs)[31] that aligns data and metadata to describe the network and its components. Heterogeneity issues in different levels, i.e., schema and data levels, across the datasets are overcome by various data curation and integration methods. Each DE comprises datasets and programs for accessing, managing, and analyzing their data. Figure 15.2 illustrates the components of the LungCancerDE Data Ecosystem. The metadata layer specifies biomedical vocabularies (e.g., Unified Medical Language System-UMLS[64] or Human Phenotype Ontology-HPO [65]). The LungCancerDE DE is a nested framework that is also composed of three basic DEs: Clinical, Scholarly, and Scientific Open. These basic DEs are described in terms of datasets, metadata, and methods; they enable each basic DEs to conduct individual analysis based on locally collected data. The nested LungCancerDE DE comprises the basic DEs and integrates the data processed by each of them. As a result, the nested LungCancerDE DE provides a holistic profile of a lung cancer patient composed of the data process by Clinical, Scholarly, and Scientific Open DEs; these profiles are represented in the LungCancerDE knowledge graph (KG). In the following, we describe the nested LungCancerDE DE in terms of the datasets of the basic DEs, and the operators, metadata, mappings, integrity constraints, and service executed on top of them.

15.3.1 Datasets

The LungCancerDE Data Ecosystem integrates three categories of data sources collected from the basic data ecosystems:

Processed Clinical Data Database produced by the Clinical Data DE as the result of the performing Natural Language Processing (NLP) on Electronic Health Records (EHR) data. EHR data correspond to 1,242 lung cancer patients registered in the Electronic Health Record (EHR) system at the Puerta del Hierro University Hospital in Madrid from 2008 to January 2020 and 15,373 lung cancer patients registered in the Electronic Health Record (EHR) system at the Spanish Lung Cancer Group (SLCG). The data is structured and presented in a relational database. Additionally, the information about the hospital services visited by the patients is shared in a relational database. The values of the attributes are in English and Spanish, and

[64]`https://www.nlm.nih.gov/research/umls/index.html`
[65]`https://hpo.jax.org/app/`

attributes like treatments are diagnostics annotated with terms from UMLS.

Scholarly Data A data graph– in Neo4J [66]– representing 162,394 scientific publications in a graph with 402,020 nodes and 12,256,983 edges. Each publication is described with a PubMed identifier, title, year, journal, authors, SCImago Journal rank indicator (sjr), Hindex, number of citations, and the link to SCOPUS with all the information of the article. Moreover, publications are annotated with 4,821,501 associations describing the relationship *has topic*, 7,368,157 associations for the relationship *mention in*, and 166,219 associations between UMLS terms.

Scientific Open Data 11,292 drugs described in terms of the conditions for which the drug can be prescribed and its interactions with targets and enzymes. There are also 60,177 relations between drugs and side effects, 1,550,586 drug-drug interactions extracted from the Literature and DrugBank, and 502,839 predicted drug-drug interactions discovered by various predictive methods. Furthermore, COSMIC coding point mutations[67] including 36.847.386 records and 38 attributes, are included. In addition, structured data related to the clinical significance of cancer genome alterations residing in Clinical Interpretation of Variants in Cancer (CIViC)[68] are also involved. This data includes diagnostic datasets, evidence of variant's impact on patient diagnosis, predictive, evidence of variant's impact on therapeutic response, predisposing, evidence of variant's role in conferring susceptibility to disease, and prognostic dataset, evidence of variant's impact on disease progression, severity, and patient survival.

15.3.2 Metadata

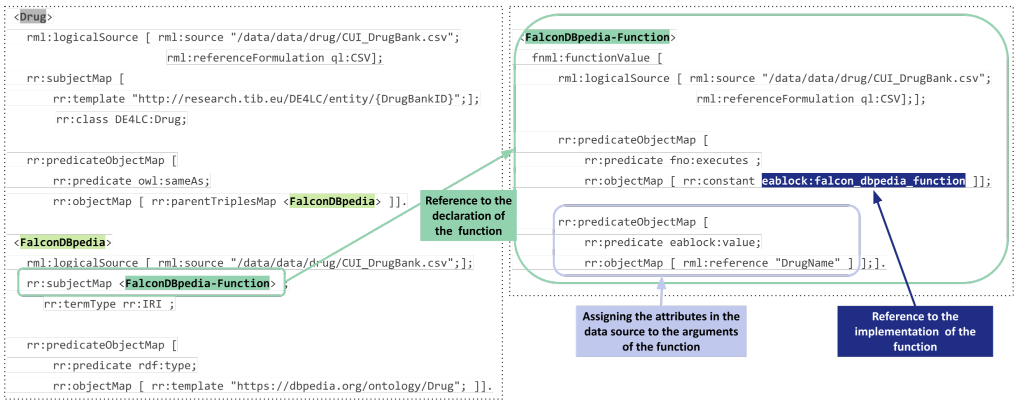
Biomedical ontologies and controlled vocabularies describe the data and provide a unified description and annotation. These annotations represent the basis of the data integration methods to merge the data into a KG. The values in the datasets are annotated with terms from the Unified Medical Language System using *EABlock*. These annotations enable entity alignment and provide the basis for integrating the datasets into the KG. A unified schema provides an integrated view of the data sources. The LungCancerDE unified schema is expressed in the W3C standard data model RDF. This increases interoperability and facilitates reusability of existing

[66] `https://neo4j.com/`

[67] `https://cancer.sanger.ac.uk/cosmicGRCh37,version90,releasedAugust2019`

[68] `https://civicdb.org/welcome`

vocabularies and ontologies, e.g., the RDF Schema[69] (RDFS), the Web Ontology Language[70] (OWL), PROV-O[71] (Provenance Ontology), Semanticscience Integrated Ontology [72] (SIO), and National Cancer Institute Thesaurus [73] (NCIT). The unified schema comprises 188 classes, 155 object properties, and 77 datatypes.



```
<Drug>
  rml:logicalSource [ rml:source "/data/data/drug/CUI_DrugBank.csv";
                      rml:referenceFormulation ql:CSV];
  rr:subjectMap [
      rr:template "http://research.tib.eu/DE4LC/entity/{DrugBankID}";];
      rr:class DE4LC:Drug;

  rr:predicateObjectMap [
      rr:predicate owl:sameAs;
      rr:objectMap [ rr:parentTriplesMap <FalconDBpedia> ]].

<FalconDBpedia>
  rml:logicalSource [ rml:source "/data/data/drug/CUI_DrugBank.csv";];
  rr:subjectMap <FalconDBpedia-Function> ;
     rr:termType rr:IRI ;

  rr:predicateObjectMap [
      rr:predicate rdf:type;
      rr:objectMap [ rr:template "https://dbpedia.org/ontology/Drug"; ]].
```

```
<FalconDBpedia-Function>
  fnml:functionValue [
      rml:logicalSource [ rml:source "/data/data/drug/CUI_DrugBank.csv";
                          rml:referenceFormulation ql:CSV];];

      rr:predicateObjectMap [
          rr:predicate fno:executes ;
          rr:objectMap [ rr:constant eablock:falcon_dbpedia_function ]];

      rr:predicateObjectMap [
          rr:predicate eablock:value;
          rr:objectMap [ rml:reference "DrugName" ] ];].
```

Reference to the declaration of the function
Assigning the attributes in the data source to the arguments of the function
Reference to the implementation of the function

Figure 15.3: An exemplary RML Mapping Rule calling one FnO function named `FalconDBpedia-Function`. This function performs NER and NEL to link drug names to resources in DBpedia.

15.3.3 Mappings

The correspondences between the data sources and the unified schema are defined using RML mapping languages. Data operation functions are defined using FnO. We apply the functions defined in *EABlock* to solve entity alignment over the textual attributes in data by aligning the recognized biomedical entities to terms in UMLS, Wikipedia [74], and DBpedia [4]. While the functions in *GenoConductor* are utilized to reconcile the genomic variant data. Figure 15.3 presents an RML where the FnO `FalconDBpedia-Function` enables the linking of drug names into resources of DBpedia. Partially, the mapping rules are defined utilizing *easyRML*. Note that the specification in RDF and the semantic description using FnO provide standard documentation of entity alignment and establish the basis for tracking down the data

[69] https://www.w3.org/TR/rdf-schema/

[70] https://www.w3.org/TR/owl-features/

[71] https://www.w3.org/TR/prov-o/

[72] https://raw.githubusercontent.com/micheldumontier/semanticscience/master/ontology/sio/release/sio-release.owl

[73] https://ncit.nci.nih.gov/ncitbrowser/

integration process. In the LungCancerDE, the combination of R2RML, RML, and FnO represents a powerful formalism to specify the pipeline for integrating data into the KG declaratively.

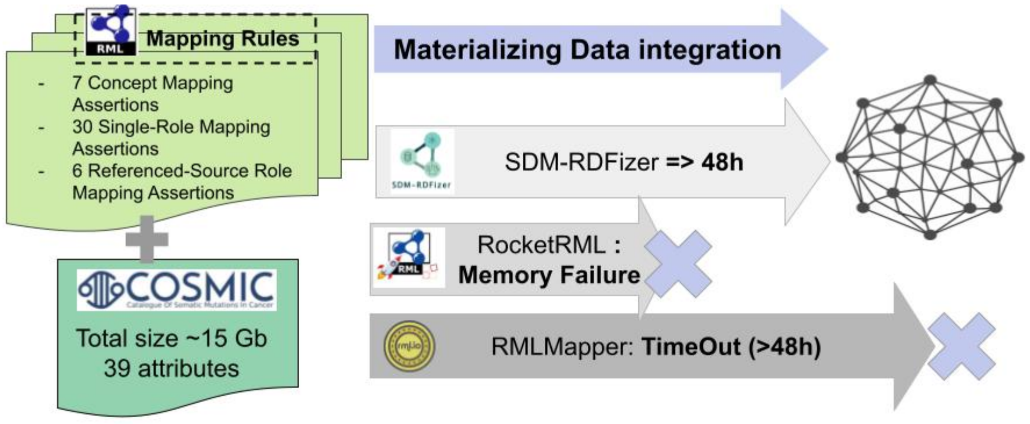

Mapping Rules
- 7 Concept Mapping Assertions
- 30 Single-Role Mapping Assertions
- 6 Referenced-Source Role Mapping Assertions
COSMIC
Total size ~15 Gb
39 attributes
Materializing Data integration
SDM-RDFizer => 48h
RocketRML : Memory Failure
RMLMapper: TimeOut (>48h)

Figure 15.4: An example of materializing a portion of LungCancerDE

15.3.4 Materializing LungCancerDE

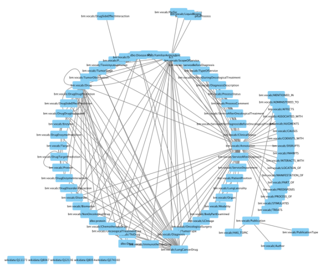
In general, in projects such as the ones that we mention in this chapter, i.e., iASiS, CLARIFY, and BigMedilytics, the task of data management is performed in parallel with data providing and extraction. This means that data integration manifested in LungCancerDE requires to be materialized regularly, based on the frequency of the data flow. Therefore, the materialization demands to be equipped with optimization techniques in order to be able to perform in a reasonable time.

Figure 15.4 illustrates an example of materializing a portion of LungCancerDE. The portion that is shown in Figure 15.4 includes only one dataset in the *Scientific Open DE* with a size of around 15 GB and the corresponding mapping rules defined in RML. Note that in this example we have included no user-defined functions. We materialized this portion of DE with different state-of-the-art frameworks including RocketRML [64], RMLMapper [22], and SDM-RDFizer [36]. We observe that RocketRML cannot create the KG because the KG creation process surpasses the memory limit of RocketRML. RMLMapper cannot generate the KG because the process took over 48 hours and timed out. SDM-RDFizer can accomplish the materialization, however, it takes about 48 hours. Considering the required time for materializing one portion of DE highlights the importance of efficiency in materializing DEs such as LungCancerDE. The demand for the practical efficient materialization of DEs in these projects makes our proposed optimization techniques desirable. Therefore, as it is shown in Figure 15.2, we utilize *MapSDI* and *Planner* techniques in order to optimize the performance of the materialization process. Furthermore, we apply *Dragoman* in materializing the correspondences between data sources and the concepts in the unified schema.

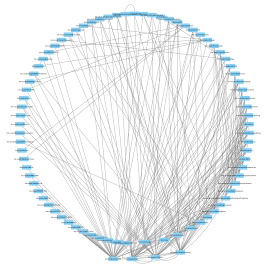
15.4 Evaluation and Results

In this section, we report some results of the contribution of our proposed techniques in four projects. Three EU-funded projects involving lung cancer pilots, i.e., iASiS, CLARIFY, and BigMedilytics, rely on the LungCancerDE explained earlier. As we interchangeably used the terms *knowledge graph creation* and *materializing data integration* in this thesis, here we explain the results of knowledge graphs that are created by applying our proposed methods.

BigMedilytics Knowledge Graph. BigMedilytics KG is evolved over the period of the project. One of the important observations during the evolution of this KG corresponds to the impact of entity alignment in data integration and improving the connectivity between the nodes in the KG. To assess the impact of entity alignment techniques that are included in LungCancerDE, we perform an empirical study to answer the following question. **Research Question.** How does applying *EABlock* in the process of KG creation impact the quality of the result KG? To answer the mentioned research question, we build two versions of BigMedilytics KG: $KG_{NoLinks}$ and KG_{Links}, the latter includes the links discovered by NER and NEL tasks applying *EABlock* data operation functions, while in the former these links have not been generated. Figure 15.5 demonstrate the network analysis results performed on the two created versions of BigMedilytics KG. These KGs are aggregated into two directed graphs $G_{NoLinks}$=$(V, E_{NoLinks})$ and G_{Links}=(V, E_{Links}). Vertices in V keep the classes in $KG_{NoLinks}$ and KG_{Links} with at least one entity. Labeled edges in $E_{NoLinks}$ (resp. E_{Links}) represent the properties that relate the entities of the classes Q and K in V, in $KG_{NoLinks}$ (resp. KG_{Links}). Thus, a labelled directed edge *e=(q,p,k)* belongs to $E_{NoLinks}$ (resp., to E_{Links}) if there are classes Q and K in V and a property p, such as q and k are instances of Q and K in V, and the RDF triple *(q, p, k)* belongs to $E_{NoLinks}$ (resp., E_{Links}). Traditional network analysis methods are conducted on top of $G_{NoLinks}$ and G_{Links} to determine connectivity. The metrics are (a) The Average number of neighbors indicates the average connectivity of a vertex or node in a graph. (b) Clustering coefficient measures the tendency of nodes who share the same



(a) $G_{NoLinks}$ without Entity Alignment



(b) G_{Links} with Entity Alignment

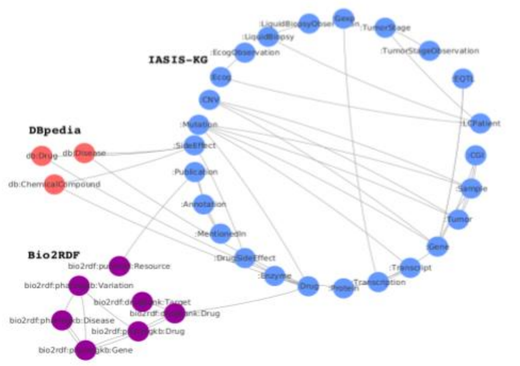
Analysis	G_{Links}	$G_{NoLinks}$
Number of nodes	82	82
Number of edges	216	180
Avg. number of neighbors	4.9	4.0
Clustering coefficient	0.13	0.09
Network density	0.03	0.02
Connected components	1	6

(c) Network Analysis

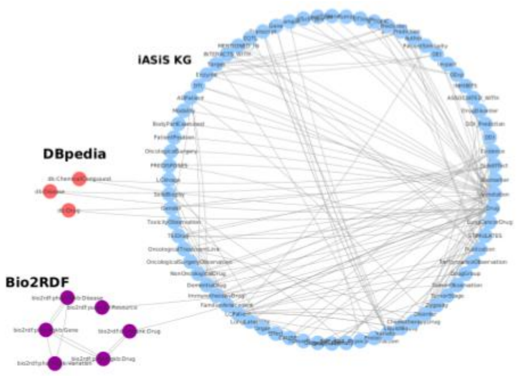
Figure 15.5: Network analysis to assess connectivity of $KG_{NoLinks}$ and KG_{Links}. Aggregated graphs $G_{NoLinks}$ and G_{Links} represent provide a summarized view of the number of connections in $KG_{NoLinks}$ and KG_{Links}.

connections in a graph to become connected. If a neighborhood is fully connected, the clustering coefficient is 1.0 while a value close to 0.0 means that there is no connection in the neighborhood. (c) Network density measures the portion of potential edges in a graph that are actually edges; a value close to 1.0 indicates that the graph is fully connected. (d) The number of connected components indicates the number of subgraphs composed of vertices connected by at least one path. Figure 15.5 depicts the aggregated graphs $G_{NoLinks}$ and G_{Links}, and Figure 15.5c reports on the results of the graph metrics. The outcomes indicate that KG_{Links} comprises more connected entities. Albeit low, the clustering coefficient and density values indicate that the UMLS annotations and links to DBpedia and Wikidata included in KG_{Links}, increase the connectivity. As a result, these connections allow for the integration into the BigMedilytics KG of the biomedical entities annotated individually in each of the data ecosystems that composed the LungCancerDE framework. Moreover, based on the results reported by Waagmeester et al. [1], which put Wikidata into perspective as a knowledge graph for the life sciences, the recognized links enrich the BigMedilytics KG with the richness of knowledge collected and maintained by the scientific communities in DBpedia and Wikidata.

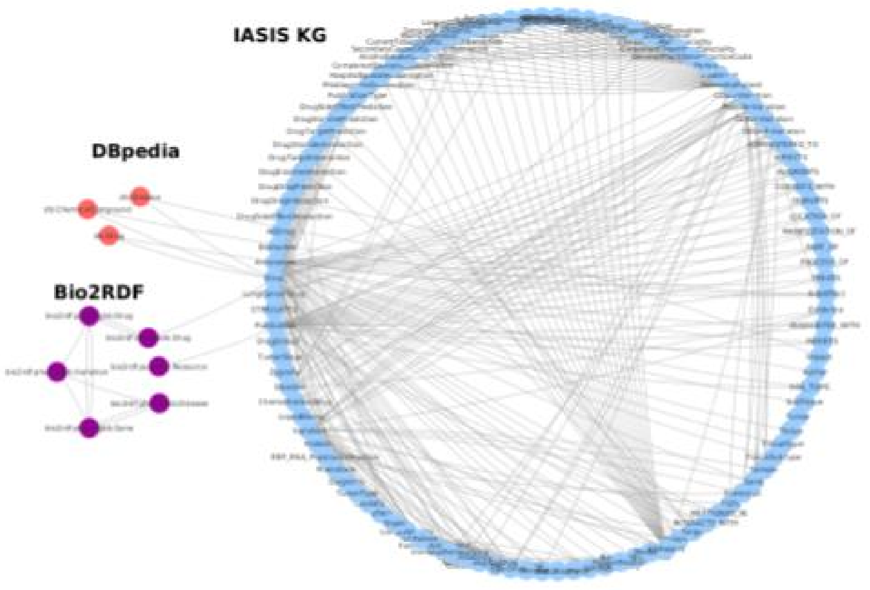
iASiS Knowledge Graph. Following the LungCancerDE and optimized materialization techniques, we created 11 versions of the iASiS knowledge graph over two years. Figure 15.6 illustrate the evolution of iASiS knowledge graph in September 2018, December 2019, and September 2020. As can be observed in Figure 15.6 the knowledge graph has grown considerably during the project.



(a) Portion of a Source File about Genes (Outer Source File)



(b) Portion of the Source File about Chromosomes (Inner Source File)



(c) September 2020

Figure 15.6: The evolution of iASiS knowledge graph over two years.

Table 15.1: An overview of the CLARIFY KG and DIS

CLARIFY	Number of Records
Data sources	13,603,943
Declared mapping assertions	19,926
Data operation functions	243
OWL classes in the unified schema	188
OWL object properties in the unified schema	155
OWL data properties in the unified schema	77
Generated RDF triples	78,194,129
KG Resources	11,424,457

CLARIFY Knowledge Graph. Since 2020, the start of project CLARIFY so far, we have created more than 10 versions of CLARIFY knowledge graph following LungCancerDE. To better understand the importance of applying our proposed optimization techniques in this project, we provide 15.1. As the number of concepts in the unified schema, mapping assertions, and data operation functions that are required in order to integrate data of this project show in 15.1, the materialization of such data integration system can be very expensive. Table 15.1 demonstrates the statistics of the current version of the CLARIFY KG (version 8.0, November 2022). As the numbers show, 11,424,457 are resources described in terms of 78,194,129 RDF triples. The resources are part of 188 classes connected by 155 directed edges. Considering, the size of data, the number of correspondence between data and classes and properties of unified schema, the considerable number of data operation functions, and finally generated RDF triples, having an efficient materialization is essential.

15.5 Summary

In this chapter, we DE4LungCancer as the computational framework to address the data management requirements of the lung cancer pilot of the H2020 EU projects iASiS, BigMedilytics, and CLARIFY. DE4LungCancer is a nested framework that comprises three DEs that process and analyzes the pilot datasets. DE4LungCancer offers a semantic layer composed of a unified schema, biomedical ontologies, and mapping languages; they provide the basis for transparent data integration into a KG. We show the necessity of optimization techniques while materializing the data integration considering the scale of data. We also illustrate exemplary results.

Chapter 16

Conclusion

In this thesis, we study the problem of efficient materialization of data integration from heterogeneous data. After defining the research questions, we follow Part I providing an overview of the essential background knowledge and the related works in Chapter 2 and Chapter 3, respectively. In Part II we explain extensively how we tackle the research problem and what we propose as solutions. This part includes six separate chapters which are followed by Part III composed of five chapters. In Part III, we introduce our testbeds and empirical studies that we perform to evaluate the solutions proposed in Part II. Lastly, Part IV starts with Chapter 15 representing the application of our proposed solutions in real-world use cases. In this final chapter, we conclude the thesis by revisiting, once again, our research questions and explaining how these questions are answered in different chapters. In closing, we review the limitations of the proposed solutions and explore future work.

16.1 Revisiting the Research Questions

Reviewing the research questions presented in Chapter 1 we explain our contributions in answering each question. Note that one research question can lead to more than one contribution and vice versa.

RQ1: What are the characteristics of a materialized data integration system?

We consider the definition of data integration systems in terms of four components including unified schema, data sources, mapping assertions, and functions.

Mapping assertions are equivalent to what is called “mapping rules” in most definitions; they represent the correspondence between the data sources and the classes and properties in the unified schema. Functions, on the other hand, represent the data operation functions, the functions for processing and manipulating data. Accordingly, in Chapter 4, we formalize the definition of mapping assertions and functions. In addition to formally defining the characteristics of materialized data integration systems involving both data operators and sources, these formalizations serve as the foundation of the following contributions.

RQ2: How to merge data operators and sources in a materialized data integration system?

The formal definitions provided in Chapter 4 pave the way to merge data operators and sources in materialized data integration systems. Relying on these formalism and available declarative mapping languages, i.e., RML+FnO, we introduce two libraries of data operation functions. The first library, *EABlock* performs entity alignment; it performs entity recognition from textual attributes and linking the recognized entities to the corresponding resources in Wikidata, DBpedia, and domain specific thesaurus, e.g., UMLS. We evaluate the impact of applying *EABlock* in data integration in improving the connections between the integrated entities in Chapter 13. The second library, *GenoConductor*, provide domain-specific data operators targeting the genomic variant harmonization. Relying on three

RQ3: How can efficiency be ensured in a materialized data integration system?

Considering the components of materialized data integration systems, we tackle the problem of efficient materialization of data integration systems from three perspectives. It should be noted that by efficiency, we aim for *optimization* in terms of *execution time*. In Chapter 11 we propose an optimization technique at the level of data sources. The optimization technique, named *MapSDI*, is composed of a set of transformation rules. *MapSDI* resorts to the properties of the relational algebra operators and to the knowledge encoded in the mapping rules to identify the transformation rules that are needed to be performed on data sources. Materializing the outcome of these transformation rules, i.e., the transformed data integration system, requires minimal execution time as empirically shown in Chapter 11, however, provides the same knowledge graphs. In Chapter 12, we study the RQ3 from *inter-mapping assertion* perspective. We introduce another optimization method, named *Planner*, relying on planning a given set of mapping assertions. *Planner* provides an optimized execution plan by partitioning and scheduling the execution of the mapping assertions. Experimental studies evaluating *Planner* explained in Chapter 12 illustrate the same results. Finally, in Chapter 9, we propose an optimization technique, named *Dragoman*, exploring the optimization opportunities at *intra-mapping assertion* level. *Dragoman*, introduces a set of mapping-based transformation rules in addition to an efficient approach for evaluating functions in mapping assertions. The materialization of the outcome of this optimization technique requires minimal time while preserving the same result as the one that can be derived from materializing the original data integration system. The results of extensive empirical evaluation of *Dragoman* is reported in Chapter 14.

RQ4: Which are the impacting parameters that testbeds need to include to evaluate the advantages of applying enhancement techniques in materialized data integration?

In Chapter 10, we start with the parameters that are reported to impact the materialization of data integration in [12]. After enumerating the parameters reported to impact the materialization of data integration, we introduce parameters that are not studied, however, they are assumed to be effective. We generate four groups of testbeds including configurations that are required to study the reported parameters. Additionally, the potential influential parameters, specifically corresponding data operation functions, are added. The results of newly studied parameters are reported in Chapter 14.

RQ5: How to overcome the challenges in applying materialized data integration systems in real-world scenarios?

In Chapter 8 we introduce *easyRML*, a tool to facilitate the process of RML mapping rules creation. *easyRML* allows users to define the rules declaring the correspondences between data sources and concepts of unified schema while exploring the data and the unified schema in real time. Furthermore, in Chapter 15 we show the practical applications of our proposed optimization techniques in three real-world projects. We introduce a data ecosystem for lung cancer as the solution of data integration in the mentioned projects. Considering the importance of efficiency in materializing data integration, we illustrate the contributions of our proposed techniques in meeting the requirements of the projects.

16.2 Limitations and Future Work

In this thesis, we focus on materialized data integration. Nevertheless, as described in Chapter 1, data can be integrated virtually. The optimization techniques proposed in this thesis are applicable to materialized data integration. For instance, the techniques proposed in Chapter 5 and Chapter 9 including transformation rules, or eager evaluation of functions are only relevant in the case of materialization. Another limitation of our proposed optimization techniques concerns the conditions according to which applying these techniques optimizes the materialization of data integrations. For instance, as explained in Subsection 14.2.4, applying our techniques proposed in *Dragoman* is more expensive materializing data integration systems which meet certain conditions. Therefore, in our future works, we investigate which of our proposed techniques are applicable to virtual data integration.

Keeping human in the loop in the process of integrating data in domains such as biomedicine is essential. Manual curation of data by domain experts and the level of expertise of them are the factors determining the credibility of data in domains like biomedicine. Additionally, following FAIR principles in research demands the application of approaches that require humans in the loop to increase precision. For instance, utilizing declarative mapping rules for defining the correspondences between data sources and concepts in the unified schema demands considerable manual effort. Our vision for future work is centered around encouraging the community to keep the human in the loop, yet, to decrease the required manual attempt. We aim to increase the reusability of existing mapping rules in defining new correspondences between the data and the unified schema. Additionally, our goal is to provide *recommendations* for knowledge engineers while defining mapping rules. To this end, we aim the work on the following contributions. 1. **Proposing different design patterns** [29]. It has been observed during different projects that similar conceptual patterns have appeared while integrating heterogeneous data. Utilizing these design patterns in designing the unified schema of data integration systems plays an important role in reusability of the generated correspondences between data sources and unified schema. 2. **Define DISs in terms of a DIS**. The associations of the

components of DISs can also be defined and formalized in terms of a DIS. Defining such DIS allow to track the associations between the components of DISs including the correlation between classes and properties of the unified schema and defined mapping rules. Possessing such correlation and reusing the design patterns allow to provide a prediction system. The prediction system enable recommendations of new mapping rules to knowledge engineers while they are defining new mapping rules.
3. **Generic data operation functions** Supplying more data operation function libraries increases the motivation in reusing the design patterns and mapping rules. Providing a spectrum of functions from being generic to specific, able to overcome heterogeneity issues due to multilinguality, and the possibility of applying them as composite functions, increase their reusability.

Appendices

Appendix A

List of Publications

1. **Samaneh Jozashoori**, Maria-Esther Vidal. *MapSDI: A scaled-up semantic data integration framework for knowledge graph creation.* In Proceedings of the 27th International Conference on Cooperative Information Systems (CooPIS), 2019 Oct 21 (pp. 58-75). Springer, Cham.

2. **Samaneh Jozashoori**, David Chaves-Fraga, Enrique Iglesias, Maria-Esther Vidal, and Oscar Corcho. *Funmap: Efficient execution of functional mappings for knowledge graph creation.* In International Semantic Web Conference, pp. 276-293. Springer, Cham, 2020.

3. **Samaneh Jozashoori**, Ahmad Sakor, Enrique Iglesias, and Maria-Esther Vidal. *EABlock: a declarative entity alignment block for knowledge graph creation pipelines.* In Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing, pp. 1908-1916. 2022.

4. **Samaneh Jozashoori**, Maria-Esther Vidal. *Data Integration for Supporting Biomedical Knowledge Graph Creation at Large-Scale.* In Proceedings of 13th International Conference, DILS 2018, Hannover, Germany, November 20-21, (2018).

5. Enrique Iglesias*, **Samaneh Jozashoori***, David Chaves-Fraga*, Diego Collarana, and Maria-Esther Vidal. *SDM-RDFizer: An RML interpreter for efficiently creating RDF knowledge graphs.* In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, pp. 3039-3046. 2020. *The first three authors contributed equally to this publication.

6. Enrique Iglesias, **Samaneh Jozashoori**, Maria-Esther Vidal. *Scaling Up*

Knowledge Graph Creation to Large and Heterogeneous Data Sources. Journal of Web Semantics, (2022).

7. Ahmad Sakor, **Samaneh Jozashoori**, Emetis Niazmand, Ariam Rivas, Kostantinos Bougiatiotis, Fotis Aisopos, Enrique Iglesias et al. *Knowledge4COVID-19: A Semantic-based Approach for Constructing a COVID-19 related Knowledge Graph from Various Sources and Analysing Treatments' Toxicities.* arXiv preprint arXiv:2206.07375 (2022).

8. Maria-Esther Vidal, **Samaneh Jozashoori**, and Ahmad Sakor. *Semantic data integration techniques for transforming big biomedical data into actionable knowledge.* In 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), pp. 563-566. IEEE, 2019.

9. Maria-Esther Vidal, Kemele M. Endris, **Samaneh Jazashoori**, Ahmad Sakor, and Ariam Rivas. *Transforming heterogeneous data into knowledge for personalized treatments—A use case.* Datenbank-Spektrum 19, no. 2 (2019): 95-106.

10. Maria-Esther Vidal, Kemele M. Endris, **Samaneh Jozashoori**, Farah Karim, and Guillermo Palma. *Semantic data integration of big biomedical data for supporting personalized medicine..* Current Trends in Semantic Web Technologies: Theory and Practice. Studies in Computational Intelligence, vol 815, pp. 25-56. Springer, Cham. (2019).

11. Maria-Esther Vidal, Kemele M. Endris, **Samaneh Jozashoori**, and Guillermo Palma. *A Knowledge-Driven Pipeline for Transforming Big Data into Actionable Knowledge.* In International Conference on Data Integration in the Life Sciences, pp. 44-49. Springer, Cham, 2018.

12. Calvo, V., E. Niazmand, E. Carcereny, **Samaneh Jozashoori**, D. Rodriguez, R. Lopez Castro, M. Guirado et al. *1730P Cancer long survivor artificial intelligence follow-up (CLARIFY): Family history of cancer and lung cancer.* Annals of Oncology 32 (2021): S1198-S1199.

Bibliography

[1] A. Waagmeester et al. “Wikidata as a knowledge graph for the life sciences”. In: *Science Forum eLife 2020;9:e52614* (2020). DOI: 10.7554/eLife.52614.

[2] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. “Foundations of modern query languages for graph databases”. In: *ACM Computing Surveys (CSUR)* 50.5 (2017), pp. 1–40.

[3] Julián Arenas-Guerrero, David Chaves-Fraga, Jhon Toledo, Marıéa S Pérez, and Oscar Corcho. “Morph-kgc: Scalable knowledge graph materialization with mapping partitions”. In: *Semantic Web* (2022).

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. “DBpedia: A Nucleus for a Web of Open Data”. In: *Proceedings of ISWC + ASWC.* 2007.

[5] João Barroca, Abhishek Shivkumar, Beatriz Quintino Ferreira, Evgeny Sherkhonov, and João Faria. “Enriching a Fashion Knowledge Graph from Product Textual Descriptions”. In: *arXiv preprint arXiv:2206.01087* (2022).

[6] Salima Benbernou, Xin Huang, and Mourad Ouziri. “Semantic-based and entity-resolution fusion to enhance quality of big rdf data”. In: *IEEE Transactions on Big Data* (2017).

[7] O. Bodenreider. “The Unified Medical Language System (UMLS): integrating biomedical terminology”. In: *Nucleic Acids Res* 1.32 (2004).

[8] Alexander Breil, Patrik Hitzelberger, Paulo Da Silva Carvalho, and Fernand Feltz. “Exploring data integration strategies for public sector cloud solutions”. In: *International Conference on Electronic Government and the Information Systems Perspective.* Springer. 2012, pp. 271–278.

[9] Dan Brickley and R.V. Guha. *RDF Schema 1.1 - W3C Recommendation 25 February 2014.* Tech. rep. Feb. 2014. URL: http://www.w3.org/TR/rdf-schema/.

[10] Cinzia Capiello, Avigdor Gal, Matthias Jarke, and Jakob Rehof. “Data ecosystems: sovereign data exchange among organizations (Dagstuhl Seminar 19391)”. In: *Dagstuhl Reports.* Vol. 9. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2020.

[11] Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. “What you always wanted to know about Datalog(and never dared to ask)”. In: *IEEE transactions on knowledge and data engineering* 1.1 (1989), pp. 146–166.

[12] David Chaves-Fraga, Kemele M Endris, Enrique Iglesias, Oscar Corcho, and Maria-Esther Vidal. “What are the parameters that affect the construction of a knowledge graph?” In: *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*. Springer. 2019, pp. 695–713.

[13] David Chaves-Fraga, Freddy Priyatna, Andrea Cimmino, Jhon Toledo, Edna Ruckhaus, and Oscar Corcho. “GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain”. In: *Journal of Web Semantics* 65 (2020), p. 100596.

[14] Alessandro Chessa, Gianni Fenu, Enrico Motta, Diego Reforgiato Recupero, Francesco Osborne, Angelo Salatino, and Luca Secchi. “Enriching Data Lakes with Knowledge Graphs”. In: ().

[15] Cuong Xuan Chu, Simon Razniewski, and Gerhard Weikum. “KnowFi: Knowledge Extraction from Long Fictional Texts”. In: *3rd Conference on Automated Knowledge Base Construction*. 2021.

[16] Diego Collarana, Mikhail Galkin, Ignacio Traverso-Ribón, Maria-Esther Vidal, Christoph Lange, and Sören Auer. “MINTE: semantically integrating RDF graphs”. In: *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*. 2017.

[17] Oscar Corcho, Freddy Priyatna, and David Chaves-Fraga. “Towards a New Generation of Ontology Based Data Access”. In: *Semantic Web Journal* 11.1 (2020).

[18] Souripriya Das, Seema Sundara, and Richard Cyganiak. “R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012”. In: *W3C* (2012).

[19] Ben De Meester, Anastasia Dimou, Ruben Verborgh, and Erik Mannens. “An ontology to semantically declare and describe functions”. In: *European Semantic Web Conference*. Springer. 2016, pp. 46–49.

[20] Ben De Meester, Wouter Maroy, Anastasia Dimou, Ruben Verborgh, and Erik Mannens. “Declarative data transformations for Linked Data generation: the case of DBpedia”. In: *European Semantic Web Conference*. 2017.

[21] Christophe Debruyne and Declan O’Sullivan. “R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings”. In: *LDOW Workshop*. 2016.

[22] Anastasia Dimou, Tom De Nies, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. “Automated Metadata Generation for Linked Data Generation and Publishing Workflows”. In: *Proceedings of the 9th Workshop on Linked Data on the Web*. Ed. by Sören Auer, Tim Berners-Lee, Christian Bizer, and Tom Heath. Vol. 1593. CEUR Workshop Proceedings. Apr. 2016.

[23] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. “RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data”. In: *Proceedings of the Workshop on Linked Data on the Web co-located with WWW*. 2014.

[24] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. “RML: a generic language for integrated RDF mappings of heterogeneous data”. In: *Ldow*. 2014.

[25] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of data integration*. Elsevier, 2012.

[26] Johan T den Dunnen, Raymond Dalgleish, Donna R Maglott, Reece K Hart, Marc S Greenblatt, Jean McGowan-Jordan, Anne-Francoise Roux, Timothy Smith, Stylianos E Antonarakis, Peter EM Taschner, et al. "HGVS recommendations for the description of sequence variants: 2016 update". In: *Human mutation* 37.6 (2016), pp. 564–569.

[27] Johan T den Dunnen and Stylianos E Antonarakis. "Mutation nomenclature extensions and suggestions to describe complex mutations: a discussion". In: *Human mutation* 15.1 (2000), pp. 7–12.

[28] Adam Frankish, Mark Diekhans, Anne-Maud Ferreira, Rory Johnson, Irwin Jungreis, Jane Loveland, Jonathan M Mudge, Cristina Sisu, James Wright, Joel Armstrong, et al. "GENCODE reference annotation for the human and mouse genomes". In: *Nucleic acids research* 47.D1 (2018).

[29] Aldo Gangemi. "Ontology design patterns for semantic web content". In: *International semantic web conference*. Springer. 2005, pp. 262–276.

[30] Gleb Gawriljuk, Andreas Harth, Craig A Knoblock, and Pedro Szekely. "A scalable approach to incrementally building knowledge graphs". In: *International Conference on Theory and Practice of Digital Libraries*. 2016.

[31] Sandra Geisler, Maria-Esther Vidal, Cinzia Cappiello, Bernadette Farias Lóscio, Avigdor Gal, Matthias Jarke, Maurizio Lenzerini, Paolo Missier, Boris Otto, Elda Paja, Barbara Pernici, and Jakob Rehof. "Knowledge-Driven Data Ecosystems Toward Data Transparency". In: *ACM J. Data Inf. Qual.* 14.1 (2022), 3:1–3:12.

[32] Thomas R Gruber. "Toward principles for the design of ontologies used for knowledge sharing?" In: *International journal of human-computer studies* 43.5-6 (1995), pp. 907–928.

[33] Claudio Gutiérrez and Juan F. Sequeda. "Knowledge graphs". In: *Commun. ACM* 64.3 (2021), pp. 96–104.

[34] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. "Knowledge graphs". In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37.

[35] Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. "Are meta-paths necessary? Revisiting heterogeneous graph embeddings". In: *Proceedings of the 27th ACM international conference on information and knowledge management*. 2018, pp. 437–446.

[36] Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, and Maria-Esther Vidal. "SDM-RDFizer: An RML interpreter for the efficient creation of rdf knowledge graphs". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 3039–3046.

[37] Enrique Iglesias, Samaneh Jozashoori, and Maria-Esther Vidal. "Scaling Up Knowledge Graph Creation to Large and Heterogeneous Data Sources". In: *Journal of Web Semantics* (2022). DOI: `https://doi.org/10.1016/j.websem.2022.100755`.

[38] Enrique Iglesias, Maria-Esther Vidal, Samaneh Jozashoori, Diego Collarana, and David Chaves-Fraga. "Empowering the SDM-RDFizer Tool for Scaling Up to Complex Knowledge Graph Creation Pipelines". In: ().

[39] Robert Isele, Anja Jentzsch, and Christian Bizer. "Silk server-adding missing links while consuming linked data". In: *Proceedings of the First International Conference on Consuming Linked Data-Volume 665*. CEUR-WS. org. 2010, pp. 85–96.

[40] Alokkumar Jha, Yasar Khan, Muntazir Mehdi, Md Rezaul Karim, Qaiser Mehmood, Achille Zappa, Dietrich Rebholz-Schuhmann, and Ratnesh Sahay. "Towards precision medicine: discovering novel gynecological cancer biomarkers and pathways using linked data". In: *Journal of biomedical semantics* 8.1 (2017), p. 40.

[41] Samaneh Jozashoori, David Chaves-Fraga, Enrique Iglesias, Maria-Esther Vidal, and Oscar Corcho. "FunMap: Efficient Execution of Functional Mappings for Knowledge Graph Creation". In: *International Semantic Web Conference*. Springer. 2020, pp. 276–293.

[42] Samaneh Jozashoori, Ahmad Sakor, Enrique Iglesias, and Maria-Esther Vidal. "EABlock: a declarative entity alignment block for knowledge graph creation pipelines". In: *Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing*. 2022, pp. 1908–1916.

[43] Samaneh Jozashoori and Maria-Esther Vidal. "MapSDI: A Scaled-Up Semantic Data Integration Framework for Knowledge Graph Creation". In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2019, pp. 58–75.

[44] Ademar Crotti Junior, Christophe Debruyne, Rob Brennan, and Declan O'Sullivan. "FunUL: a method to incorporate functions into uplift mapping languages". In: *Intern. Confer. on Information Integration and Web-based Applications and Services*. 2016.

[45] Craig A Knoblock and Pedro Szekely. "Exploiting Semantics for Big Data Integration." In: *AI Magazine* 36.1 (2015).

[46] Benjamin Lang, Alexandros Armaos, and Gian G Tartaglia. "RNAct: Protein–RNA interaction predictions for model organisms with supporting experimental data". In: *Nucleic acids research* 47.D1 (2018).

[47] Maurizio Lenzerini. "Data integration: A theoretical perspective". In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2002.

[48] Benjamin Lewin, Jocelyn Krebs, Stephen T. Kilpatrick, and Elliott S. Goldstein. *Lewin's GENES X*. 2011.

[49] Deborah L McGuinness, Frank Van Harmelen, et al. "OWL web ontology language overview". In: *W3C recommendation* 10.10 (2004), p. 2004.

[50] Pablo N Mendes, Hannes Mühleisen, and Christian Bizer. "Sieve: linked data quality assessment and fusion". In: *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. 2012, pp. 116–123.

[51] Manuel Namici and Giuseppe De Giacomo. "Comparing Query Answering in OBDA Tools over W3C-Compliant Specifications." In: *Description Logics*. 2018.

[52] Axel-Cyrille Ngonga Ngomo and Sören Auer. "LIMES—a time-efficient approach for large-scale link discovery on the web of data". In: *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[53] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. "Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done". In: *Queue* 17.2 (2019), pp. 48–75.

[54] Marcelo Iury S Oliveira and Bernadette Farias Lóscio. "What is a data ecosystem?" In: *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*. 2018, pp. 1–9.

[55] *RMLMapper: v4.12*. `https://github.com/RMLio/rmlmapper-java`. Accessed: 24-06-2022.

[56] Ahmad Sakor, Samaneh Jozashoori, Emetis Niazmand, Ariam Rivas, Kostantinos Bougiatiotis, Fotis Aisopos, Enrique Iglesias, Philipp D Rohde, Trupti Padiya, Anastasia Krithara, et al. "Knowledge4COVID-19: A semantic-based approach for constructing a COVID-19 related knowledge graph from various sources and analysing treatments' toxicities". In: *Journal of Web Semantics* (2022), p. 100760.

[57] Ahmad Sakor, Kuldeep Singh, Anery Patel, and Maria-Esther Vidal. "Falcon 2.0: An Entity and Relation Linking Tool over Wikidata". In: *International Conference on Information and Knowledge Management (CIKM)*. 2020.

[58] Andreas Schultz, Andrea Matteini, Robert Isele, Christian Bizer, and Christian Becker. "LDIF-linked data integration framework". In: *Proceedings of the Second International Conference on Consuming Linked Data-Volume 782*. 2011, pp. 125–130.

[59] *SDM-Genomic Testbeds*. `https://doi.org/10.6084/m9.figshare.17142371`. Accessed: 24-06-2022.

[60] *SDM-RDFizer: v3.6*. `https://pypi.org/project/rdfizer/3.6/`. Accessed: 24-06-2022.

[61] *SDM-RDFizer: v4.0*. `https://pypi.org/project/rdfizer/4.0/`. Accessed: 24-06-2022.

[62] Abraham Silberschatz, Henry F Korth, Shashank Sudarshan, et al. *Database system concepts*. Vol. 4. McGraw-Hill New York, 1997.

[63] Umutcan Şimşek, Elias Kärle, and Dieter Fensel. *RocketRML - A NodeJS implementation of a use-case specific RML mapper*. Accessed: 24-06-2022. 2019. arXiv: `1903.04969 [cs.SE]`.

[64] Umutcan Şimşek, Elias Kärle, and Dieter Fensel. "RocketRML-A NodeJS implementation of a use-case specific RML mapper". In: *Proceeding of the First International Workshop on Knowledge Graph Building*. 2019.

[65] Pedro Szekely, Craig A Knoblock, Jason Slepicka, Andrew Philpot, Amandeep Singh, Chengye Yin, Dipsy Kapoor, Prem Natarajan, Daniel Marcu, Kevin Knight, et al. "Building and using a knowledge graph to combat human trafficking". In: *International Semantic Web Conference*. 2015.

[66] Vicenç Torra. "Lazy and Eager Evaluation". In: *Scala: From a Functional Programming Perspective*. Springer, 2016, pp. 37–50.

[67] *Tracemalloc: Version 3.4*. `https://docs.python.org/3/library/tracemalloc.html`. Accessed: 24-06-2022. 2020.

[68] Hensin Tsao, Lynda Chin, Levi A Garraway, and David E Fisher. "Melanoma: from mutations to medicine". In: *Genes & development* 26.11 (2012), pp. 1131–1155.

[69] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume II*. Computer Science Press, 1989.

[70] 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* 19.2 (2019), pp. 95–106.

[71] Maria-Esther Vidal, Kemele M Endris, Samaneh Jozashoori, and Guillermo Palma. "A Knowledge-Driven Pipeline for Transforming Big Data into Actionable Knowledge". In: *International Conference on Data Integration in the Life Sciences*. Springer. 2018, pp. 44–49.

[72] Maria-Esther Vidal, Samaneh Jozashoori, and Ahmad Sakor. "Semantic data integration techniques for transforming big biomedical data into actionable knowledge". In: *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE. 2019, pp. 563–566.

[73] Maria-Esther Vidal, Ahmad Sakor, Samaneh Jozashoori, Emetis Niazmand, Disha Purohit, Enrique Iglesias, Fotis Aisopos, Dimitrios Vogiatzis, Ernestina Menasalvas, Alejandro Rodriguez Gonzalez, et al. "Knowledge Graphs for Enhancing Transparency in Health Data Ecosystems". In: ().

[74] D. Vrandecic and M. Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Commun. ACM* 57.10 (2014), pp. 78–85.

[75] Binh Vu, Jay Pujara, and Craig A Knoblock. "D-REPR: A Language for Describing and Mapping Diversely-Structured Data Sources to RDF". In: *Intern. Confer. on Knowledge Capture*. 2019.

[76] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific data* 3.1 (2016), pp. 1–9.

[77] David Wood, Markus Lanthaler, and Richard Cyganiak. *RDF 1.1 Concepts and Abstract Syntax*. Feb. 2014. URL: `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/` (visited on 03/15/2015).

[78] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. "Virtual knowledge graphs: An overview of systems and use cases". In: *Data Intelligence* 1.3 (2019), pp. 201–223.

[79] Jennifer L Yen, Sarah Garcia, Aldrin Montana, Jason Harris, Stephen Chervitz, Massimo Morra, John West, Richard Chen, and Deanna M Church. "A variant by any name: quantifying annotation discordance across tools and clinical databases". In: *Genome medicine* 9.1 (2017), p. 7.