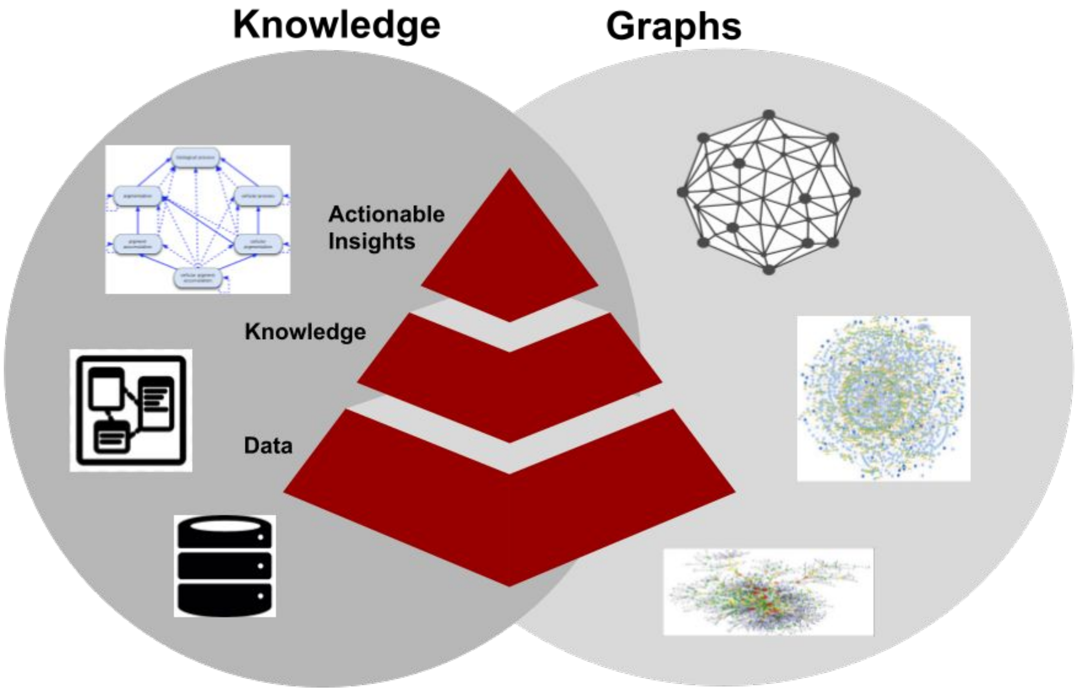
Challenges for Efficiently Creating and Maintaining Knowledge Graphs

Prof. Dr. Maria-Esther Vidal
Scientific Data Management Group

Why do we need Knowledge Graphs?

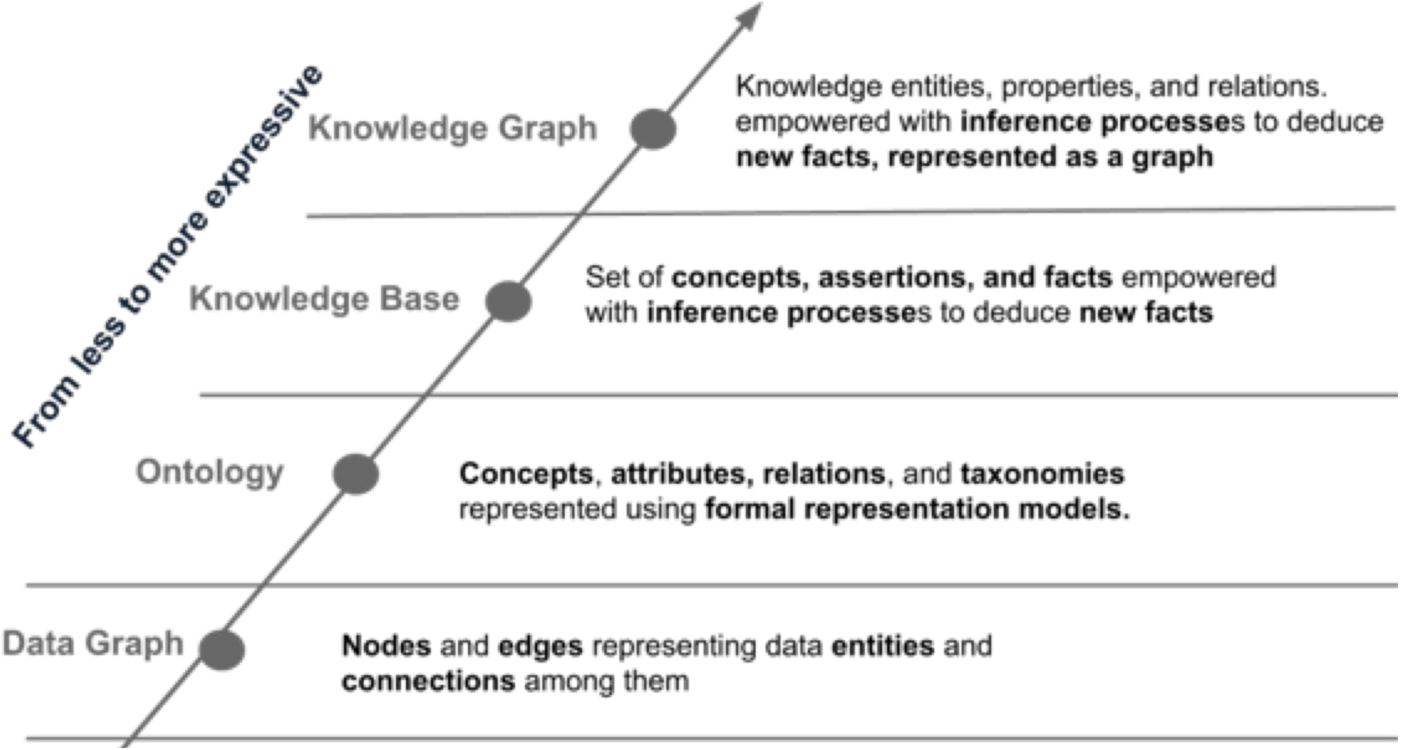
Knowledge Graphs



Knowledge graphs

- data **structures**
- represent the **convergence** of **knowledge** and **data** as **factual** statements
- use a **graph data model**

Spectrum of Knowledge Graphs

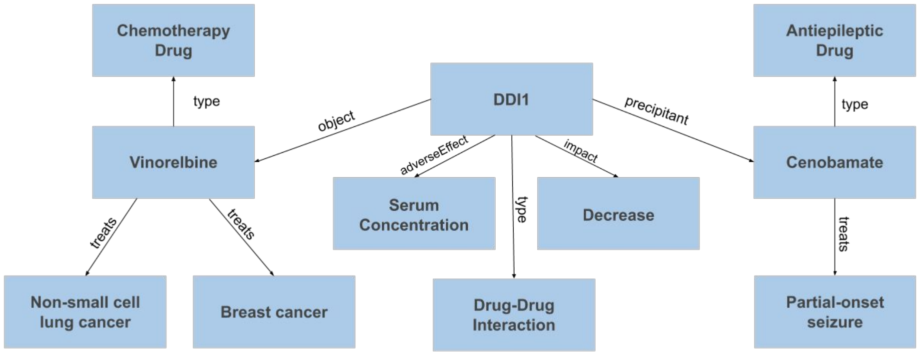


An Example of Knowledge Graphs

Vinorelbine is a chemotherapy drug that is used in the treatment of breast cancer and non-small cell lung cancer (NSCLC).

Cenobamate is an antiepileptic drug used to treat partial-onset seizures.

The serum concentration of Vinorelbine can be decreased when it is combined with Cenobamate.

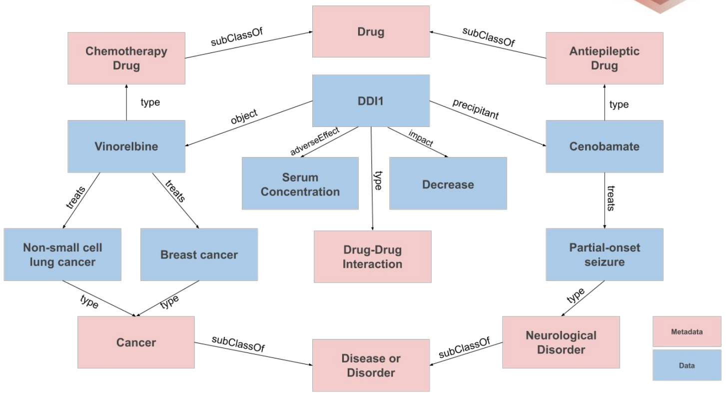


Entities and relationships are **first-class citizens** and **representation** of relationships between **entities**

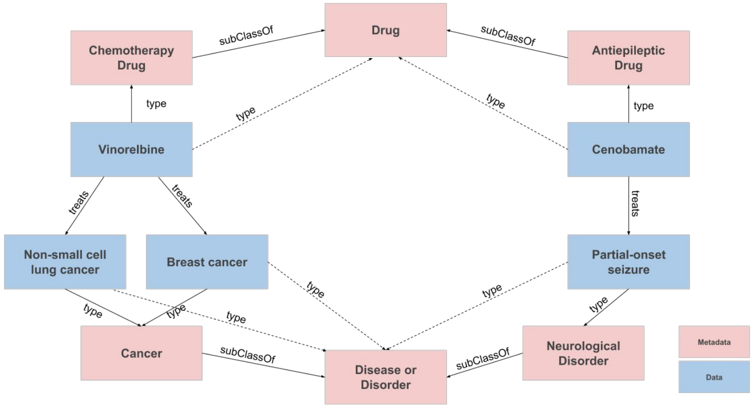
Properties of Knowledge Graphs

Natural representation of metadata

- Meaning of **entities** and **relationships**



Metadata and **data** can be **empowered** with **inference processes** to deduce **new facts**



Knowledge Graphs: Benefits and Challenges

Knowledge graphs

- Provide a **formal specification** of the **meaning** of entities
  - **Metadata**: data describing and providing information about other data
- Model **taxonomies** of **entities**, **relationships**, and **classes**
- Develop a **common understanding** of a **domain**
- Natural **representation** of **metadata**
  - **Meaning** of **entities** and **relationships**
- **Metadata** and **data** can be **empowered** with **inference** to deduce new **facts**

Analysis on top of Knowledge Graphs

Lung Cancer Protocols:

Afatinib is a second generation Tyrosine Kinase Inhibitors (TKI) **not** recommended for non-small cell lung cancer patients with **Epidermal Growth Factor Receptor (EGFR)** mutation negative.

Lapatinib is a dual Tyrosine Kinase Inhibitors (TKI) for non-small cell lung cancer patients with **HER2** mutation positive or EGFR positive.

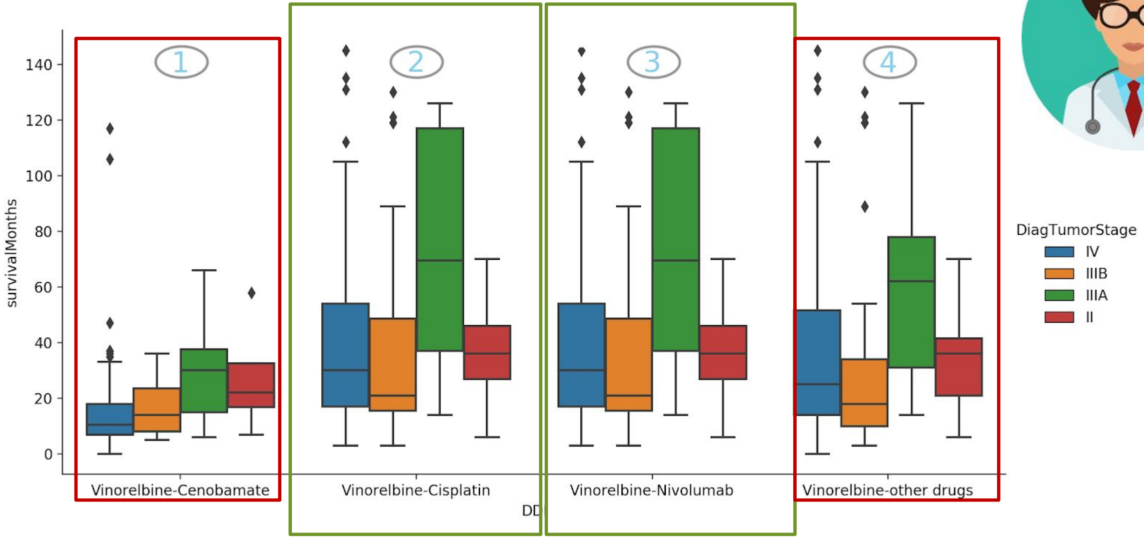
Is this an error in the **data** stored in the **KG** or did **ex:patient1** receive **this treatment**?

Instances of a knowledge graph:

```
ex:patient1 rdf:type ex:NSCLG-EGFR-negative .
ex:patient1 rdf:type ex:NSCLG-HER2-OR-EGFR-positive .
ex:patient1 ex:hasOncologicalTreatment dbpedia:Afatinib .
```

Analysis on top of Knowledge Graphs

How can support a trustable validation of the reported analysis?



Survival Analysis of **non-small lung cancer patients** categorized by **tumor stage** and different **oncological treatments** in **combination** with **Vinorelbine**

Case 1: Vinorelbine and **Cenobamate** may interact.

Case 2: Vinorelbine and **Cisplatin** interact, but are there further studies that report the effectiveness of them?

Case 3: Vinorelbine and **Nivolumab** cannot be prescribed together. This must be an error!

Case 4: Are there further studies that support the effectivenes of **Vinorelbine?**

Challenges for tracing data integrated in Knowledge Graphs

Case 1: Vinorelbine and **Cenobamate** r
interact.

DRUGBANK

DRUG	INTERACTION
Celecoxib	The metabolism of Celecoxib can be decreased when combined with Vinorelbine.
Celiprolol	The metabolism of Celiprolol can be decreased when combined with Vinorelbine.
Cenobamate	The serum concentration of Vinorelbine can be decreased when it is combined with Cenobamate.

Human readable representation.

Interactions between your drugs

Drugs.com
Know more. Be sure.

Moderate **vinorelbine ⟨ ⟩ cenobamate**
Applies to: Navelbine (vinorelbine) and cenobamate

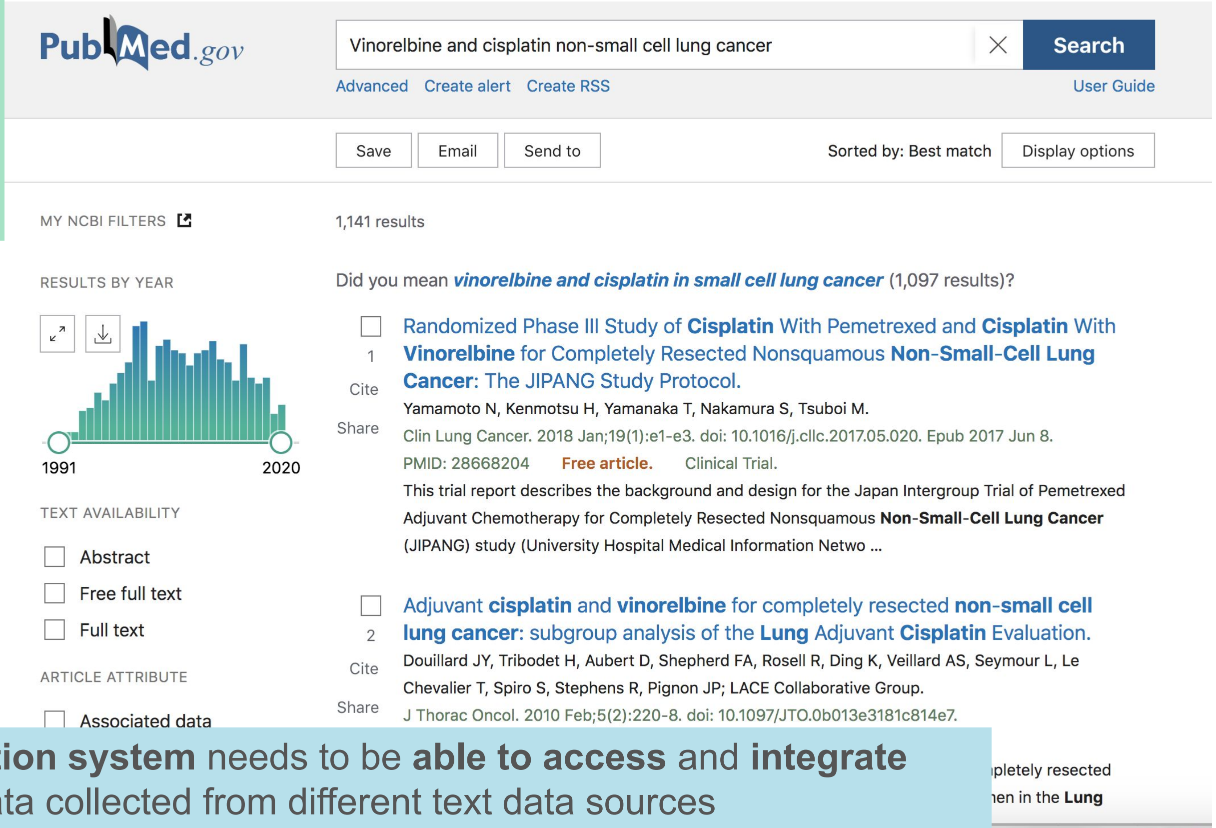
Cenobamate may reduce the blood levels of vinorelbine, which may make the medication less effective in treating your condition. Talk to your doctor if you have any questions or concerns. Your doctor may be able to prescribe alternatives that do not interact, or you may need a dose adjustment or more frequent monitoring to safely use both medications. It is important to tell your doctor about all other medications you use, including vitamins and herbs. Do not stop using any medication without first talking to your doctor.

A **data integration system** needs to be **able to access** and **integrate** unstructured data collected from different text data sources

Challenges for tracing data integrated in Knowledge Graphs

Case 2: Vinorelbine and **Cisplatin** interact, but are there further studies that report the effectiveness of them?

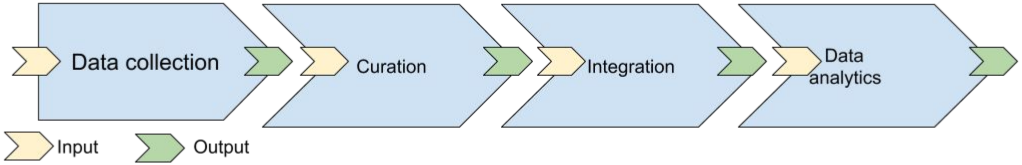
Case 4: Are there further studies that support the effectivenes of **Vinorelbine?**

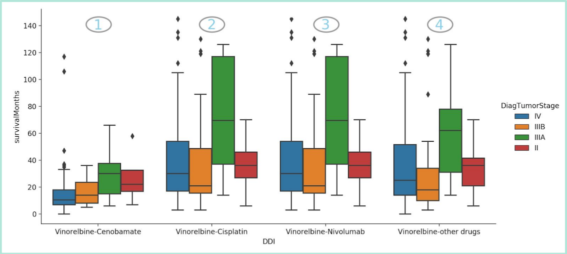


A **data integration system** needs to be **able to access** and **integrate** unstructured data collected from different text data sources

Challenges for tracing data integrated in Knowledge Graphs

Case 3: Vinorelbine and **Nivolumab** cannot be prescribed together. This must be an error!





Data transparency requires **tracking down** all the steps of the data-driven pipeline

- **accounting** for the decisions made by each component of the pipeline
- **describing** of raw data and quality issues present in the raw data sets
- **validating** of clinical data to verify if there are patients that take Vinorelbine and Nivolumab together
- **certifying** that data protection regulations are respected in all the steps!

Knowledge Graphs: Benefits and Challenges

Knowledge graphs

- Provide a **formal specification** of the **meaning** of entities
  - **Metadata**: data describing and providing information about other data
- Model **taxonomies** of **entities**, **relationships**, and **classes**
- Develop a **common understanding** of a **domain**
- Natural **representation** of **metadata**
  - **Meaning** of **entities** and **relationships**
- **Metadata** and **data** can be **empowered** with **inference** to deduce new **facts**

Data Integration

- Natural Language and Image Processing for **recognizing relevant entities**
- Techniques for **entity linking** ... **ent**
- Data **quality** assessm... **ation**

Knowledge Re... **on**

- Ex... **alism** and **reasoning** ...isms
- ...ethods for **integrity constraint** validation

Computationally Expensive in Time and Space

Knowledge Discovery

- Methods able to **discover** patterns in knowledge graphs

Predictive Models

- Capable to exploit the **semantics** encoded in knowledge graphs towards **explainable AI**

Agenda

1. Data Integration Systems, Data Ecosystems, and Knowledge Graphs
2. Declarative Mapping Languages
3. Evaluation of Pipelines for Knowledge Graph Creation
4. Integrity Constraint Validation
5. Pipelines for KG Creation
6. Future Directions

Data Integration Systems, Data Ecosystems, and Knowledge Graphs

3646

Mediators in the Architecture of Future Information Systems

Gio Wiederhold
Stanford University

September 1991

An edited version of this report was published in
The IEEE Computer Magazine, March 1992

Abstract

The installation of high-speed networks using optical fiber and high bandwidth messsage forwarding gateways is changing the physical capabilities of information systems. These capabilities must be complemented with corresponding software systems advances to obtain a real benefit. Without smart software we will gain access to more data, but not improve access to the type and quality of information needed for decision making.

To develop the concepts needed for future information systems we model information processing as an interaction of data and knowledge. This model provides criteria for a high-level functional partitioning. These partitions are mapped into information processing modules. The modules are assigned to nodes of the distributed information systems. A central role is assigned to modules that *mediate* between the users' workstations and data resources. Mediators contain the administrative and technical knowledge to create information needed for decision-making. Software which mediates is common today, but the structure, the interfaces, and implementations vary greatly, so that automation of integration is awkward.

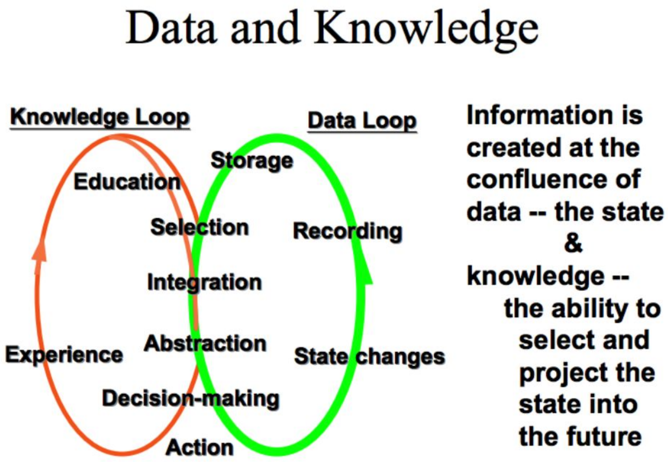
By formalizing and implementing mediation we establish a partitioned information systems architecture, which is of managable complexity and can deliver much of the power that technology puts into our reach. The partitions and modules map into the powerful distributed hardware that is becoming available. We refer to the modules that perform these services in a sharable and composable way as *mediators*.

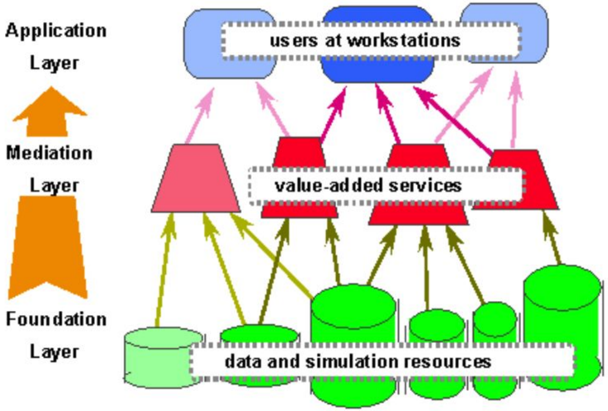
We will present conceptual requirements that must be placed on mediators to assure effective large-scale information systems. The modularity in this architecture is not only a goal, but also enables the goal to be reached, since these systems will need autonomous modules to permit growth and enable them to survive in a rapidly changing world.

The intent of this paper is to provide a conceptual framework for many distinct efforts. The concepts provide a direction for an information processing systems in the foreseeable

1

Data, Knowledge and Interoperability [Wiederhold'92]

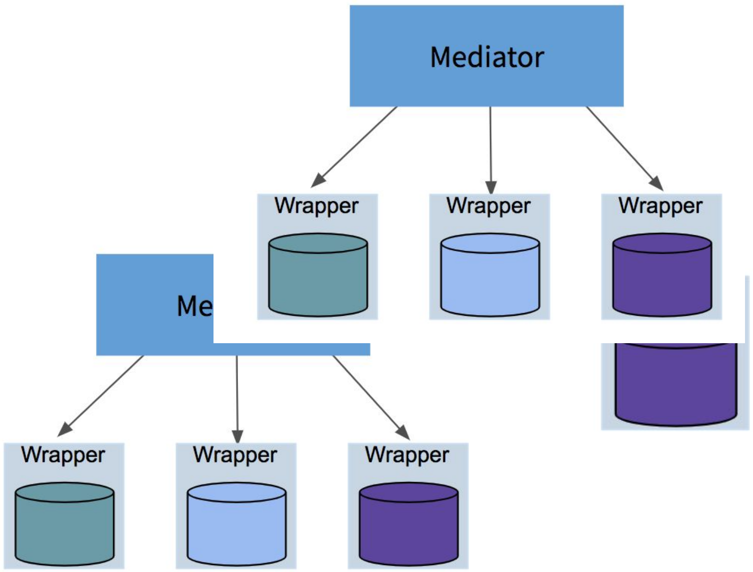




Data describes specific instances and events. Data may gathered automatically or clerically. The correctness of data can be verified vis-a-vis the real world.

Knowledge describes abstract classes. Each class typically covers many instances. Experts are needed to gather and formalize knowledge. Data can be used to disprove knowledge.

Mediator and Wrapper Architecture [Wiederhold'92]



Wrappers:

- provide access to heterogeneous data sources. For each data
- export information about source schema, data, and query processing capabilities.

Mediators:

- store the information provided by wrappers in a unified view of all available data with a central data dictionary
- decompose input queries into sub-queries that can be executed by wrappers
- gather results from wrappers and create answers to the user query

3646

Mediators in the Architecture of Future Information Systems

Gio Wiederhold
Stanford University
September 1991
An edited version of this report was published in
The IEEE Computer Magazine, March 1992

Abstract

The installation of high-speed networks using optical fiber and high bandwidth messsage forwarding gateways is changing the physical capabilities of information systems. These capabilities must be complemented with corresponding software systems advances to obtain a real benefit. Without smart software we will gain access to more data, but not improve access to the type and quality of information needed for decision making.

To develop the concepts needed for future information systems we model information processing as an interaction of data and knowledge. This model provides criteria for a high-level functional partitioning. These partitions are mapped into information processing modules. The modules are assigned to nodes of the distributed information systems. A central role is assigned to modules that *mediate* between the users' workstations and data resources. Mediators contain the administrative and technical knowledge to create information needed for decision-making. Software which mediates is common today, but the structure, the interfaces, and implementations vary greatly, so that automation of integration is awkward.

By formalizing and implementing mediation we establish a partitioned information systems architecture, which is of managable complexity and can deliver much of the power that technology puts into our reach. The partitions and modules map into the powerful distributed hardware that is becoming available. We refer to the modules that perform these services in a sharable and composable way as *mediators*.

We will present conceptual requirements that must be placed on mediators to assure effective large-scale information systems. The modularity in this architecture is not only a goal, but also enables the goal to be reached, since these systems will need autonomous modules to permit growth and enable them to survive in a rapidly changing world.

The intent of this paper is to provide a conceptual framework for many distinct efforts. The concepts provide a direction for an information processing systems in the foreseeable

1

3569

Data Integration: A Theoretical Perspective

Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lenzerini@dis.uniroma1.it

ABSTRACT

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data. The problem of designing data integration systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view. This document presents on overview of the material to be presented in a tutorial on data integration. The tutorial is focused on some of the theoretical issues that are relevant for data integration. Special attention will be devoted to the following aspects: modeling a data integration application, processing queries in data integration, dealing with inconsistent data sources, and reasoning on queries.

1. INTRODUCTION

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [60, 61, 89]. The problem of designing data integration systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view. This tutorial is focused on some of these theoretical issues, with special emphasis on the following topics.

The data integration systems we are interested in this work are characterized by an architecture based on a global schema and a set of sources. The sources contain the real data, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. Modeling the relation between the sources and the global schema is therefore a crucial aspect. Two basic approaches have been proposed to this purpose. The first approach, called global-as-view, requires that the global schema is expressed in terms of the data sources. The second approach, called local-as-view, requires the global schema to be specified independently from the sources, and the relationships between the global schema and the sources are established by defining every source as a view over the global schema. Our goal is to discuss the characteristics of the two modeling mechanisms, and to mention other possible approaches.

Irrespectively of the method used for the specification of the mapping between the global schema and the sources, one basic service provided by the data integration system is to answer queries posed in terms of the global schema. Given the architecture of the system, query processing in data integration requires a reformulation step: the query over the global schema has to be reformulated in terms of a set of queries over the sources. In this tutorial, such a reformulation problem will be analyzed for both the case of local-as-view, and the case of global-as-view mappings. A main theme will be the strong relationship between query processing in data integration and the problem of query answering with incomplete information.

Since sources are in general autonomous, in many real-world applications the problem arises of mutually inconsistent data sources. In practice, this problem is generally dealt with by means of suitable transformation and cleaning procedures applied to data retrieved from the sources. In this tutorial, we address this issue from a more theoretical perspective.

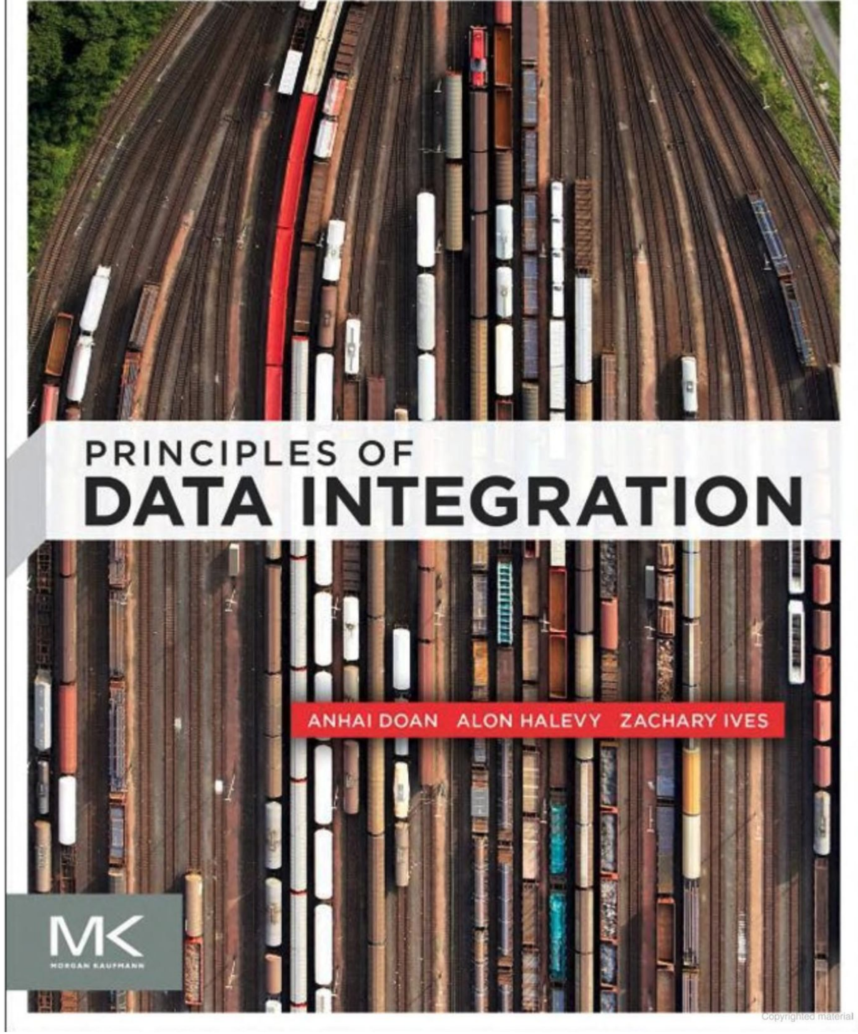
Finally, there are several tasks in the operation of a data integration system where the problem of reasoning on queries (e.g., checking whether two queries are equivalent) is relevant. Indeed, query containment is one of the basic problems in database theory, and we will discuss several notions generalizing this problem to a data integration setting.

The paper is organized as follows. Section 2 presents our formalization of a data integration system. In Section 3 we discuss the various approaches to modeling. Sections 4 and 5 present an overview of the methods for processing queries in the local-as-view and in the global-as-view approach, respectively. Section 6 discusses the problem of dealing with inconsistent sources. Section 7 provides an overview on the problem of reasoning on queries. Finally, Section 8 concludes the paper by mentioning some open problems, and several research issues related to data integration that are not addressed in the tutorial.

2. DATA INTEGRATION FRAMEWORK

In this section we set up a logical framework for data integration. We restrict our attention to data integration systems

Permission to make digital or hard copies of part or all of this work or personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
ACM PODS 2002 June 3-6, Madison, Wisconsin, USA
© 2002 ACM 1-58113-507-6/02/06...$5.00.



2012

https://www.google.de/books/edition/Principles_of_Data_Integration/5Rg679tjhFQC?hl=en&gbpv=1

Data Integration Systems[Lenzerini2002]

DIS=<O,S,M>

Let O be a set of general concepts in a general schema or ontology.

Let S={S1,..,Sn} be a set of symbols representing data sources.

Let M be a set of mappings between data sources in S and general concepts in O.

Data Integration Systems- Paradigms [Lenzerini2002]

Global-as-View (GAV):

- Concepts in the Global Schema (O) are defined in terms of combinations of Sources (S).

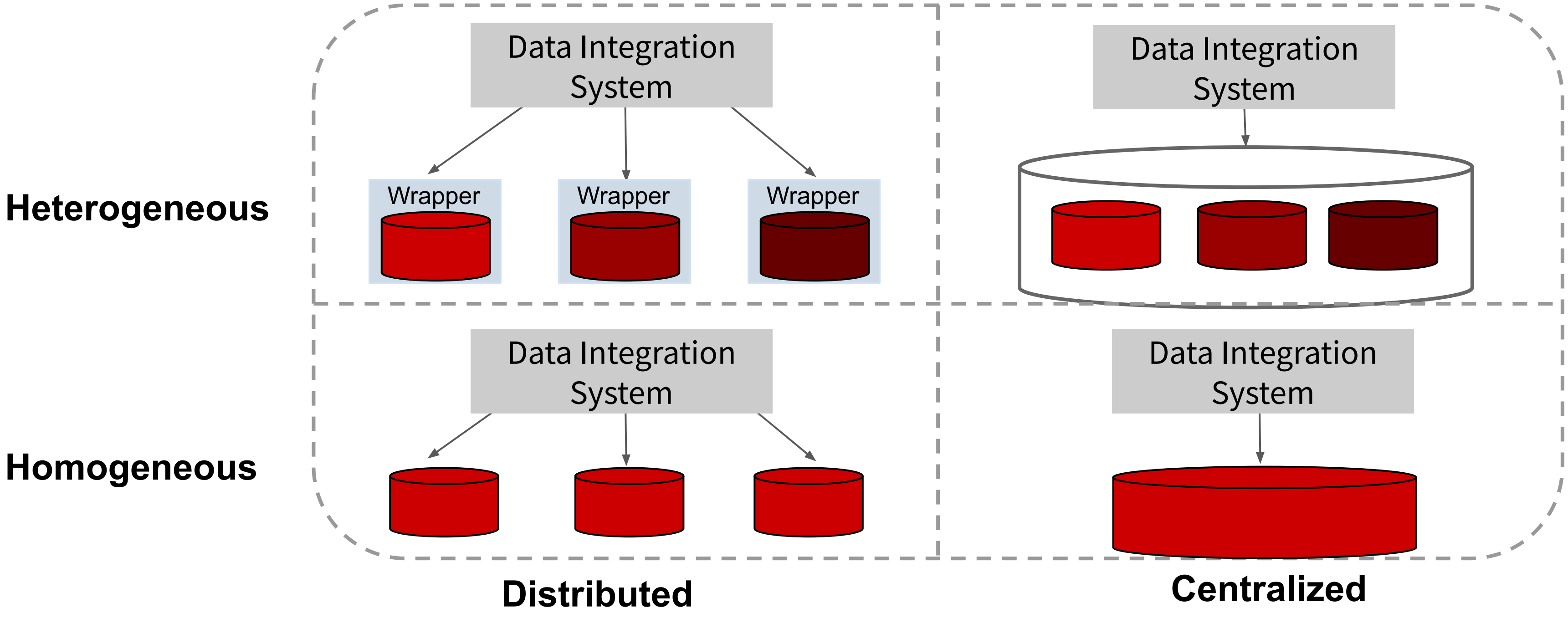
Local-As-View (LAV):

- Sources in S are defined in terms of combinations of Concepts in O.

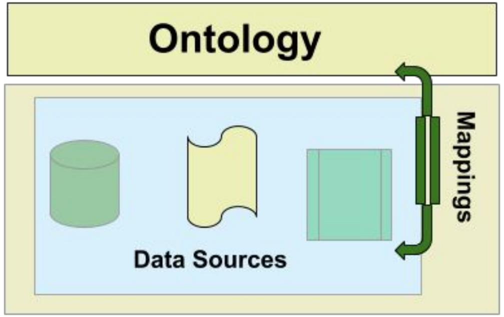
Global- & Local-As-View (GLAV):

- Combinations of concepts in the Global Schema (O) are defined in combinations of Sources (S).

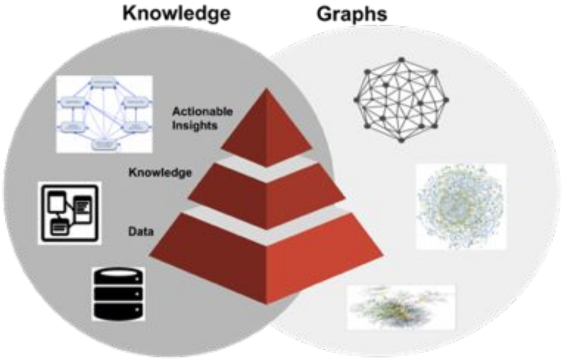
Data Integration Systems



Knowledge Graphs



Materialized Data Integration System



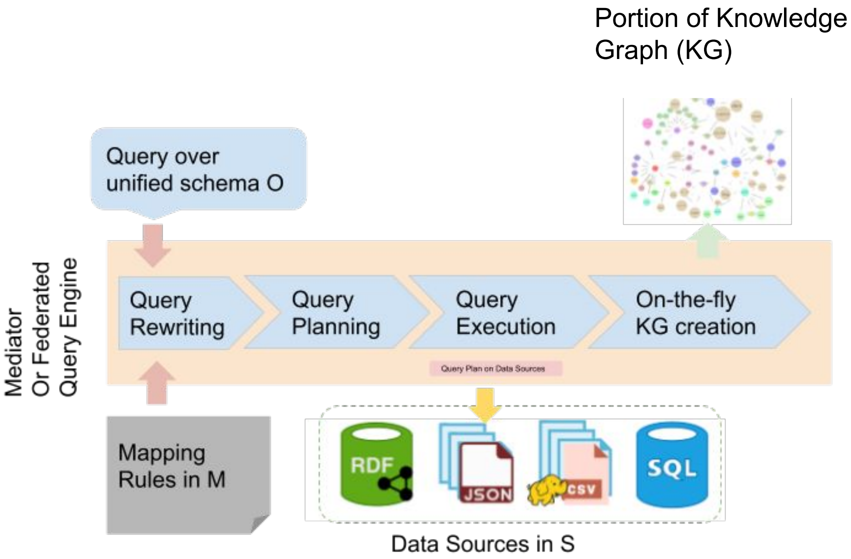
A **Knowledge Graph** is a graph **KG=(O,V,E)**:

- **O** is a unified schema
- **V** is a set of entities representing data, information, or knowledge. Types of the entities in **V** are defined in **O**
- **E** is a set of edges between entities in **V. Edges** are labeled with predicates in **O**. The semantics of these predicates is also stated in **O.**

Vidal M.E., et al. Transforming Heterogeneous Data into Knowledge for Personalized Treatments - A Use Case. Datenbank-Spektrum 19(2):(2019)
Geisler S., Vidal M-E, et al. Knowledge-driven Data Ecosystems Towards Transparency. ACM Journal Data and Information Quality. 2021

Virtual Data Integration Systems

- **Mapping rules** in M are used to rewrite a query *Q* over unified schema O into a query *Q'* in the data sources in S
- **Query planning** is performed to optimized *Q'* and generate a query plan *QP* on the data sources
- **Query execution engine** evaluates *QP* in the selected data sources
- **Query answers** are used to create a portion of the Knowledge Graph

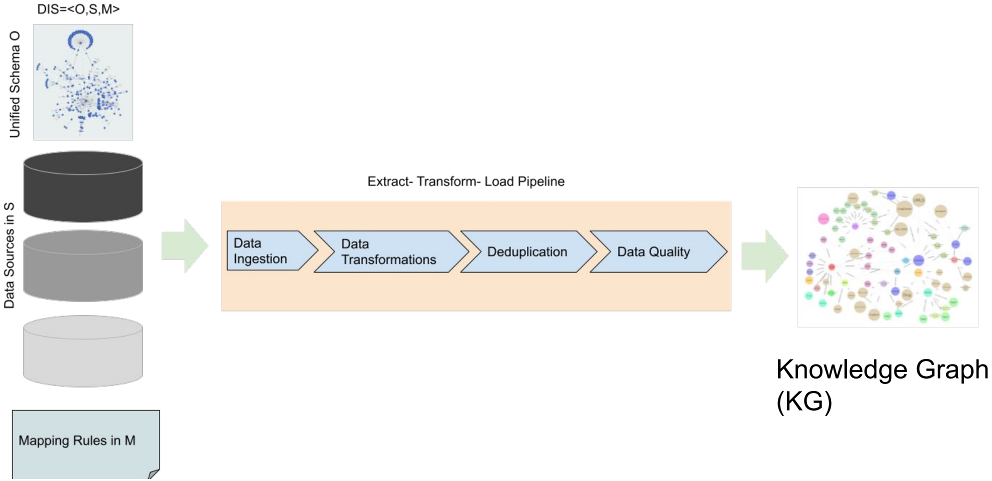


Virtual Data Integration Systems

Ontop (Calvanese et al.) https://ontop-vkg.org/	● Creates a virtual RDF KG during the evaluation of SPARQL against relational databases. ● Mapping rules are specified in R2RML ● Ontology specified in RDFS or OWL QL
Ultrawrap (Sequeda and Miranker) https://www.cs.utexas.edu/~miranker/studentWeb/UltrawrapHomePage2.html	● The unified schema O is created from the SQL DDL of a relational database ● The relational tables are represented as triples using views (virtual triple store) ● Transform every SPARQL query against the O into SQL on the virtual triple views
Morph (Priyatna et al.) https://github.com/fpriyatna/morph	● Creates RDF KGs from relational data sources based on a SPARQL query and R2RML mapping rules ● Employs query execution techniques to generate efficient SQL queries against the relational databases
Ontario (Endris et al.) https://github.com/SDM-TIB/Ontario	● Executes SPARQL queries against data sources in various formats ○ XML, relational databases, JSON ● Implements query execution techniques to generate efficient query plans
Morph-CSV (Chaves et al.)	● Enhances the process SPARQL-SQL over tabular data (defined as CSV files) with domain-specific constraints ● Implements query execution techniques to generate efficient query plans

Materialized Data Integration Systems

- Data sources are integrated as instances of the unified schema O
- Mapping rules in M are executed to generate the unified schema O instances
- Controlled vocabularies are utilized for data annotation as basis for entity alignment
- Usually implemented by Extraction-Transform-Load (ETL) tools



Materialzed Data Integration Systems

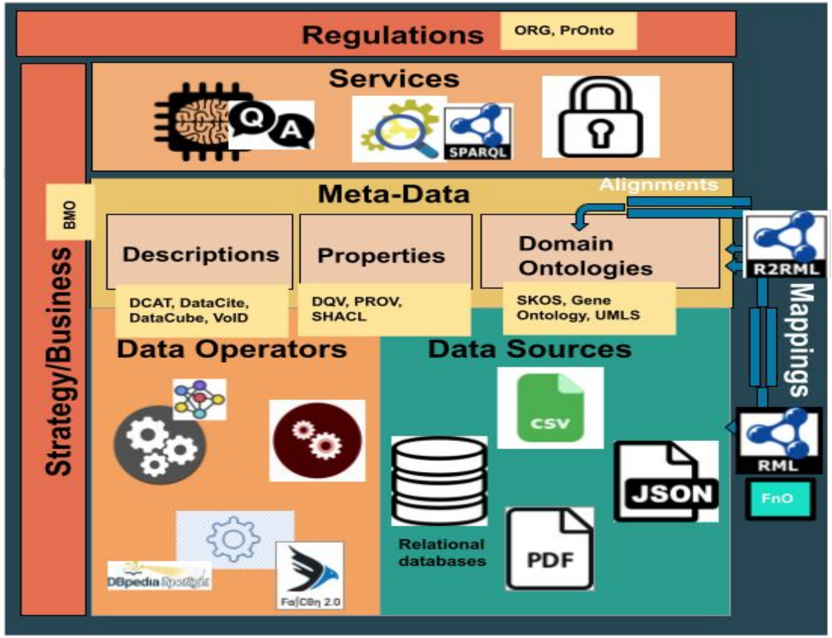
RMLMapper (Dimou et al. 2016) https://github.com/RMLio/rmlmapper-java	● In-memory engine. RML compliant engine to create RDF graphs from data sources in various formats ○ local formats: CSV, JSON.XML, Excel file, LibreOffice ○ remote access: SPARQL endpoints, Web APIs, relational databases ● Provides drivers to access multiple types of data sources ○ Oracle,, MySQL,PostgreSQL, SQLServer, and WebAPIs
SDM-RDFizer (Iglesias et at. 2020) https://github.com/SDM-TIB/SDM-RDFizer	● RML compliant engine able to transform data into RDF ○ local formats: CSV, JSON.XML ○ remote access: relational databases ● Implement data structures and physical operators to efficiently execute RML mapping rules ● Produces results incrementally ● Able to trace down the execution of RML mapping rules
Morph-KGC (Arenas-Guerrero et al.) https://github.com/oeg-upm/morph-kgc	● RML compliant engine able to transform data into RDF ○ local formats: CSV, JSON.XML ○ remote access: relational databases ● Implement planning techniques for planning mappings ○ based on mapping partitioning

Materialzed Data Integration Systems

TIB

Rocket-RML (Şimşek et al.) https://semantifyit.github.io/RocketRML/	● RML compliant engine ○ XML, JSON, CSV ● Tuned to work with large XML or JSON files
SPARQL-Generate (Lefrançois et al.**)** https://ci.mines-stetienne.fr/sparql-generate/	● Extends SPARQL 1.1 binding function mechanism to ○ query and iterate over data streams in various formats ■ RDF, SQL, XML, JSON, CSV, GeoJSON, WebSocket streams, Web APIs ○ transform the collected data using SPARQL 1.1 functions and operators ○ populate instances in an RDF based on Graph-pattern templates
Chimera (Scrocca et al.**)** https://github.com/cefriel/chimera	● Generic pipeline for RDF graph creation and configurable for RML ● Implements optimization techniques for managing large JSON and XML files ● Plans the execution of mapping rules to reduce memory consumption ● Generates RDF triples incrementally and upload them in a triple store

A Knowledge-driven Data Ecosystem



Data Ecosystems: distributed, open, and adaptive information systems with the characteristics of being self-organized, scalable, and sustainable.

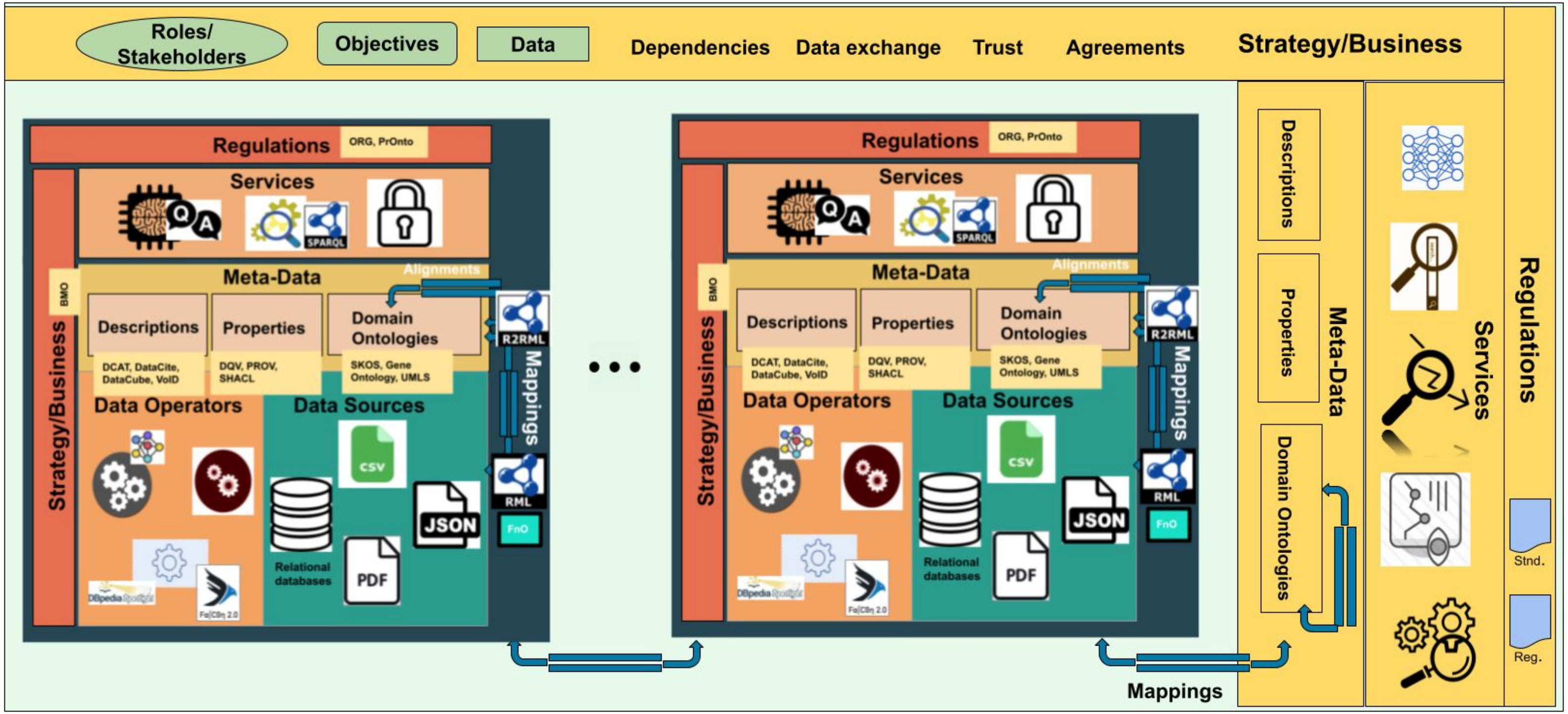
Data Operators: are functions used for accessing or managing data in the data sets.
Domain ontologies: provide unified views of the concepts, relationships, and constraints of the domain of knowledge.
Properties: enable the definition of data quality, provenance, and data access regulations of the data.
Descriptions: characteristics of data sources using standards and controlled vocabularies
Mappings: correspondences among the different components.

Services able to exploit **knowledge encoded in the metadata** to support **transparency and traceability**

- Question answering, query processing, data integration, entity and predicate linking, and data quality validation

Geisler S., Vidal M-E, et al. Knowledge-driven Data Ecosystems Towards Transparency. ACM Journal Data and Information Quality. 2022

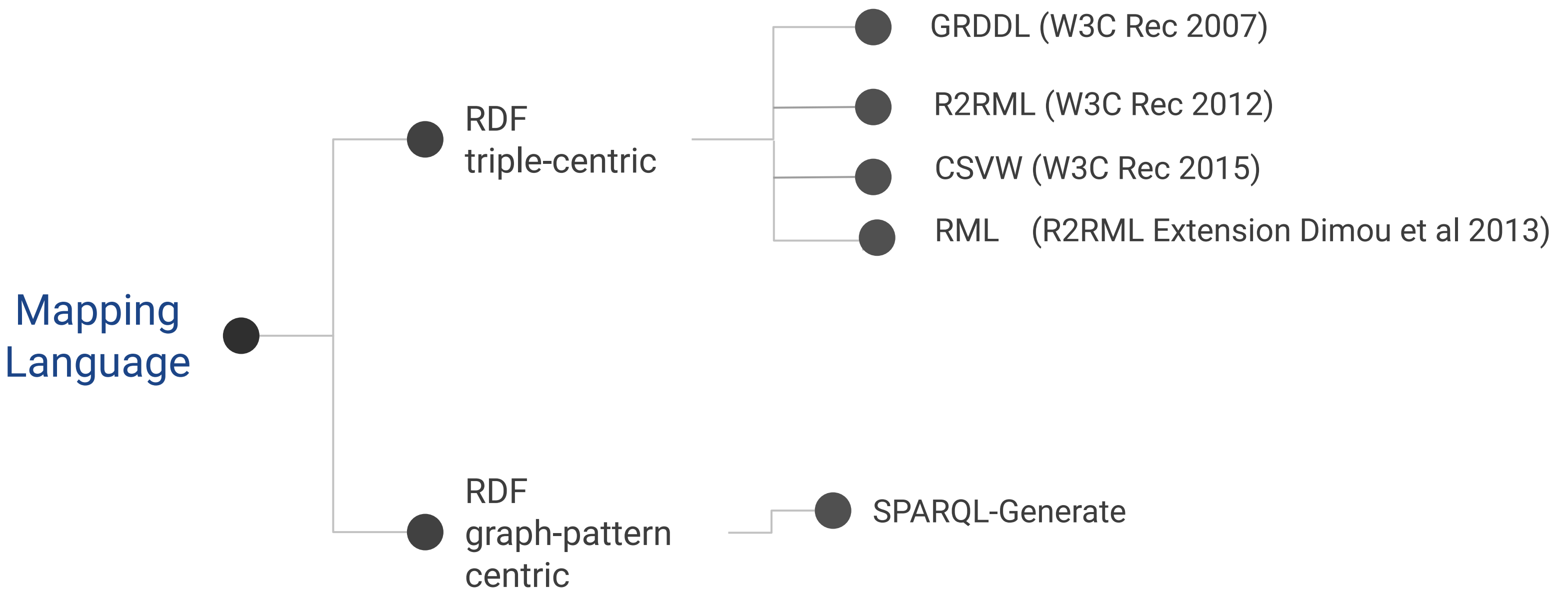
Network of Knowledge-driven Data Ecosystems



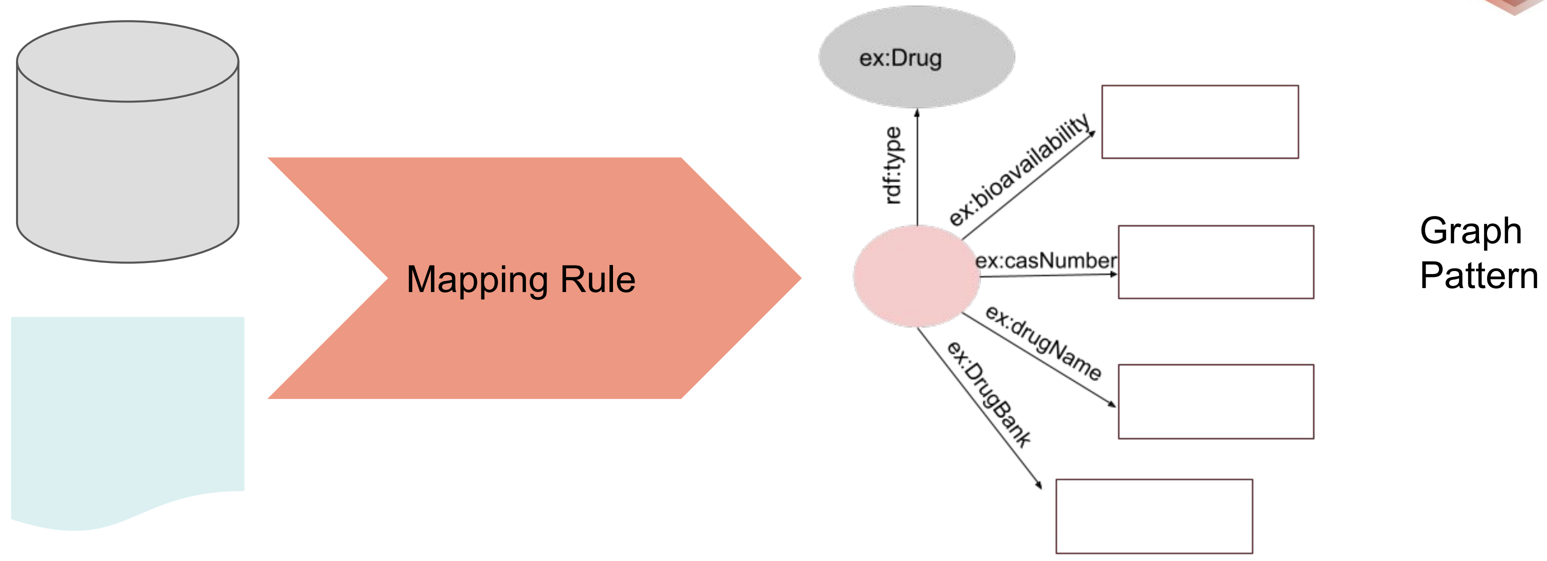
Geisler S., Vidal M-E, et al. Knowledge-driven Data Ecosystems Towards Transparency. ACM Journal Data and Information Quality. 2022

Declarative Mapping Languages

Declarative Mapping Languages



Graph-Pattern Centric Mapping Languages



The language enables the definition of mapping rules

- **gather** data from **various data sources** and transform the **collected data** into **instances of a graph pattern**

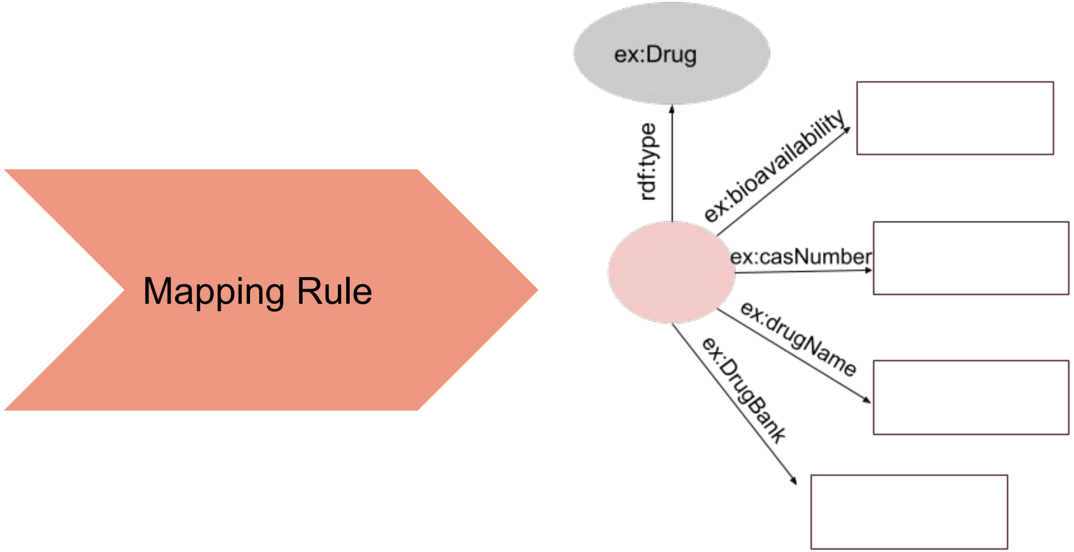
SPARQL-Generate: implementable on top of existing SPARQL engines

Heterogeneous Data Sources

https://www.slideshare.net/maximelefrancois86/overview-of-the-sparqlgenerate-language-and-latest-developments

SPARQL-Generate - Graph-Pattern Centric Mapping Language

```
{"DrugDescription":
[ {"DrugName": "Vinorelbine",
   "Bioavailability": "43.000000",
   "casNumber": "71486-22-1",
   "drugbankID": "DB00361"},

  {"DrugName": "Cisplatine",
   "Bioavailability": "100.000000",
   "casNumber": "15663-27-1",
   "drugbankID": "DB00515"},

  {"DrugName": "Omeprazole",
   "Bioavailability": "35.000000",
   "casNumber": "73590-58-6",
   "drugbankID": "DB00338"},
]
}
```



SPARQL-Generate - Generate Query

```
{“DrugDescription”:
[ {“DrugName”: “Vinorelbine”,
   “Bioavailability”: “43.000000”,
   “casNumber”: “71486-22-1”,
   “drugbankID”: “DB00361”},

  {“DrugName”: “Cisplatine”,
   “Bioavailability”: “100.000000”
   “casNumber”: “15663-27-1”,
   “drugbankID”: “DB00515”},

  {“DrugName”: “Omeprazole”,
   “Bioavailability”: “35.000000”,
   “casNumber”: “73590-58-6”,
   “drugbankID”: “DB00338”},
]
}
```

```
PREFIX fun: <http://w3id.org/sparql-generate/fn/>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
GENERATE {
[] a ex:Drug;
   ex:bioavailability ?bio;
   ex:casNumber ?number;
   ex:drugName ?name;
   ex:DrugBank ?ID.
}
SOURCE <drugDocument> as ?document
ITERATOR iter:JSONPath(?document, "$.DrugDescription.[*]") AS ?drug
WHERE {
  BIND(fun:JSONPath(?drug, "$.DrugName") AS ?name)
  BIND(fun:JSONPath(?drug, "$.Bioavailability") AS ?bio)
  BIND(fun:JSONPath(?drug, "$.casNumber") AS ?number)
  BIND(fun:JSONPath(?drug, "$.drugbankID") AS ?ID)
}
```

SPARQL-Generate - Graph-Pattern Template

{“DrugDescription”:
[{“DrugName”: “Vinorelbine”,
“Bioavailability”: “43.000000”,
“casNumber”: “71486-22-1”,
“drugbankID”: “DB00361”},

{“DrugName”: “Cisplatine”,
“Bioavailability”: “100.000000”
“casNumber”: “15663-27-1”,
“drugbankID”: “DB00515”},

{“DrugName”: “Omeprazole”,
“Bioavailability”: “35.000000”,
“casNumber”: “73590-58-6”,
“drugbankID”: “DB00338”},
]
}

```
PREFIX fun: <http://w3id.org/sparql-generate/fn/>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
GENERATE {
[] a ex:Drug;
   ex:bioavailability ?bio;
   ex:casNumber ?number;
   ex:drugName ?name;
   ex:DrugBank ?ID.
}
SOURCE <drugDocument> as ?document
ITERATOR iter:JSONPath(?document, "$.DrugDescription.[*]") AS ?drug
WHERE {
  BIND(fun:JSONPath(?drug, "$.DrugName") AS ?name)
  BIND(fun:JSONPath(?drug, "$.Bioavailability") AS ?bio)
  BIND(fun:JSONPath(?drug, "$.casNumber") AS ?number)
  BIND(fun:JSONPath(?drug, "$.drugbankID") AS ?ID)
}
```

SPARQL-Generate - Data Source

```
{“DrugDescription”:
[ {“DrugName”: “Vinorelbine”,
   “Bioavailability”: “43.000000”,
   “casNumber”: “71486-22-1”,
   “drugbankID”: “DB00361”},

  {“DrugName”: “Cisplatine”,
   “Bioavailability”: “100.000000”
   “casNumber”: “15663-27-1”,
   “drugbankID”: “DB00515”},

  {“DrugName”: “Omeprazole”,
   “Bioavailability”: “35.000000”,
   “casNumber”: “73590-58-6”,
   “drugbankID”: “DB00338”},
]
}
```

```
PREFIX fun: <http://w3id.org/sparql-generate/fn/>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
GENERATE {
[] a ex:Drug;
   ex:bioavailability ?bio;
   ex:casNumber ?number;
   ex:drugName ?name;
   ex:DrugBank ?ID.
}
SOURCE <drugDocument> as ?document
ITERATOR iter:JSONPath(?document, "$.DrugDescription.[*]") AS ?drug
WHERE {
  BIND(fun:JSONPath(?drug, "$.DrugName") AS ?name)
  BIND(fun:JSONPath(?drug, "$.Bioavailability") AS ?bio)
  BIND(fun:JSONPath(?drug, "$.casNumber") AS ?number)
  BIND(fun:JSONPath(?drug, "$.drugbankID") AS ?ID)
}
```

SPARQL-Generate - Iterator for source traversal

{“DrugDescription”:
[{“DrugName”: “Vinorelbine”,
“Bioavailability”: “43.000000”,
“casNumber”: “71486-22-1”,
“drugbankID”: “DB00361”},

{“DrugName”: “Cisplatine”,
“Bioavailability”: “100.000000”
“casNumber”: “15663-27-1”,
“drugbankID”: “DB00515”},

{“DrugName”: “Omeprazole”,
“Bioavailability”: “35.000000”,
“casNumber”: “73590-58-6”,
“drugbankID”: “DB00338”},
]
}

```
PREFIX fun: <http://w3id.org/sparql-generate/fn/>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
GENERATE {
[] a ex:Drug;
   ex:bioavailability ?bio;
   ex:casNumber ?number;
   ex:drugName ?name;
   ex:DrugBank ?ID.
}
SOURCE <drugDocument> as ?document
ITERATOR iter:JSONPath(?document, "$.DrugDescription.[*]") AS ?drug
WHERE {
  BIND(fun:JSONPath(?drug, "$.DrugName") AS ?name)
  BIND(fun:JSONPath(?drug, "$.Bioavailability") AS ?bio)
  BIND(fun:JSONPath(?drug, "$.casNumber") AS ?number)
  BIND(fun:JSONPath(?drug, "$.drugbankID") AS ?ID)
}
```

SPARQL-Generate - Binding functions

```
{“DrugDescription”:
[ {“DrugName”: “Vinorelbine”,
   “Bioavailability”: “43.000000”,
   “casNumber”: “71486-22-1”,
   “drugbankID”: “DB00361”},

  {“DrugName”: “Cisplatine”,
   “Bioavailability”: “100.000000”
   “casNumber”: “15663-27-1”,
   “drugbankID”: “DB00515”},

  {“DrugName”: “Omeprazole”,
   “Bioavailability”: “35.000000”,
   “casNumber”: “73590-58-6”,
   “drugbankID”: “DB00338”},
]
}
```

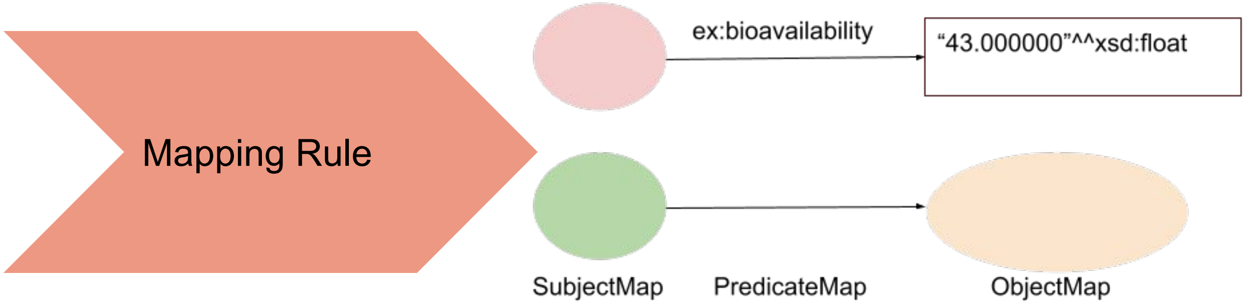
```
PREFIX fun: <http://w3id.org/sparql-generate/fn/>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
GENERATE {
[] a ex:Drug;
   ex:bioavailability ?bio;
   ex:casNumber ?number;
   ex:drugName ?name;
   ex:DrugBank ?ID.
}
SOURCE <drugDocument> as ?document
ITERATOR iter:JSONPath(?document, "$.DrugDescription.[*]") AS ?drug
WHERE {
  BIND(fun:JSONPath(?drug, "$.DrugName") AS ?name)
  BIND(fun:JSONPath(?drug, "$.Bioavailability") AS ?bio)
  BIND(fun:JSONPath(?drug, "$.casNumber") AS ?number)
  BIND(fun:JSONPath(?drug, "$.drugbankID") AS ?ID)
}
```

RDF Triple Centric Mapping Language

{“DrugDescription”:
[{“DrugName”: “Vinorelbine”,
“Bioavailability”: “43.000000”,
“casNumber”: “71486-22-1”,
“drugbankID”: “DB00361”},

{“DrugName”: “Cisplatine”,
“Bioavailability”: “100.000000”,
“casNumber”: “15663-27-1”,
“drugbankID”: “DB00515”},

{“DrugName”: “Omeprazole”,
“Bioavailability”: “35.000000”,
“casNumber”: “73590-58-6”,
“drugbankID”: “DB00338”},
]
}



R2RML

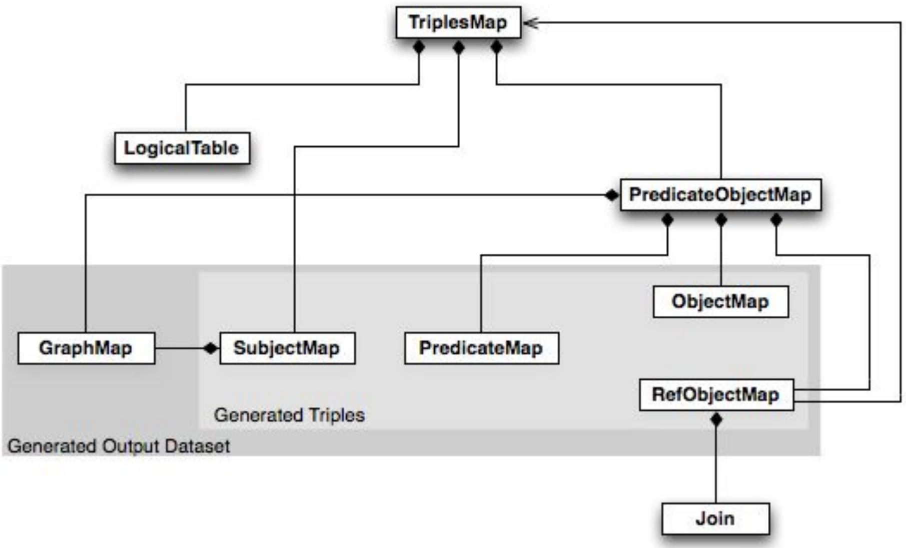
Mapping rules (**Triples maps**) from relational tables into RDF (**LogicalTable**)

- ○ a base table
- ○ a view
- ○ a valid SQL query

A **Subject Map** generates the subject of RDF triples

Predicate-Object Maps assign predicate and object to a subject

- **predicate Map** indicates the predicate
- **object Map** defines the object

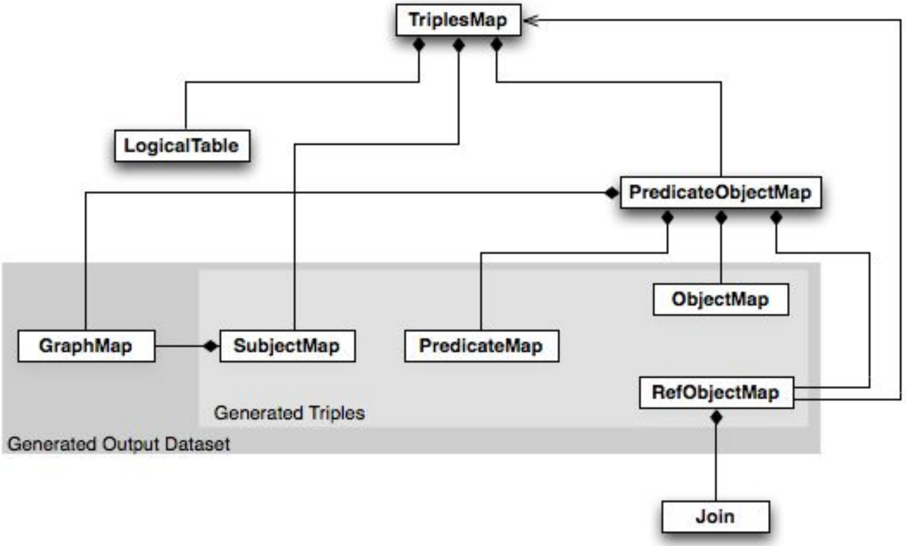


https://www.w3.org/TR/r2rml/

R2RML

RefObject Maps allows for the definition of the values of an object as the subjects of the RDF triples generated by another **TriplesMap**

Join indicates the condition to be satisfied to retrieve the **subject values** of the referenced **triples map**

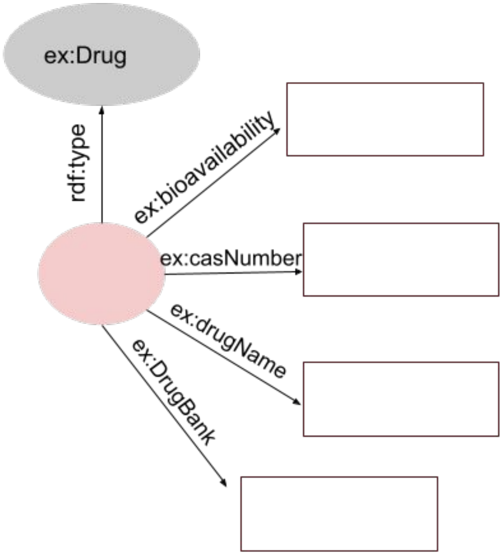


https://www.w3.org/TR/r2rml/

R2RML

Relational Table- Drug

DrugName	Bioavailability	CasNumber	DrugBankID
Vinorelbine	43.000000	71486-22-1	DB00361
Cisplatine	100.000000	15663-27-1	DB00515
Omeprazole	35.000000	73590-58-6	DB00338



Triples Map

R2RML- Logical Table

DrugName	Bioavailability	CasNumber	DrugBankID
Vinorelbine	43.000000	71486-22-1	DB00361
Cisplatine	100.000000	15663-27-1	DB00515
Omeprazole	35.000000	73590-58-6	DB00338

```
@prefix rr: <http:www.w3,org/ns/r2rml#>.
@prefix ex: <http://example.com/>.
<TriplesMap>
    rr:logicalTable [rr:tableName "Drug"];
    rr:subjectMap [
        rr:template "http://example.com/Drug/{DrugBankID}";
        rr: class ex:Drug;];

    rr:predicateObjectMap [
        rr:predicate ex:biovailability ;
        rr:objectMap [rr:column "Bioavailability"; rr:datatype xsd:float];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:casNumber ;
        rr:objectMap [rr:column "CasNumber"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:drugName ;
        rr:objectMap [rr:column "DrugName"; rr:language "en-us"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:DrugBank;
        rr:objectMap [rr:column "DrugBankID"];
    ].
```

R2RML- SubjectMap- PredicateObject Map

DrugName	Bioavailability	CasNumber	DrugBankID
Vinorelbine	43.000000	71486-22-1	DB00361
Cisplatine	100.000000	15663-27-1	DB00515
Omeprazole	35.000000	73590-58-6	DB00338

```
@prefix rr: <http:www.w3,org/ns/r2rml#>.
@prefix ex: <http://example.com/>.
<TriplesMap>
    rr:logicalTable [rr:tableName "Drug"];
    rr:subjectMap [
        rr:template "http://example.com/Drug/{DrugBankID}";
        rr: class ex:Drug;];

    rr:predicateObjectMap [
        rr:predicate ex:biovailability ;
        rr:objectMap [rr:column "Bioavailability"; rr:datatype xsd:float];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:casNumber ;
        rr:objectMap [rr:column "CasNumber"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:drugName ;
        rr:objectMap [rr:column "DrugName"; rr:language "en-us"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:DrugBank;
        rr:objectMap [rr:column "DrugBankID"];
    ].
```

R2RML- Logical Table- SQL Query

DrugName	Bioavailability	CasNumber	DrugBankID
Vinorelbine	43.000000	71486-22-1	DB00361
Cisplatine	100.000000	15663-27-1	DB00515
Omeprazole	35.000000	73590-58-6	DB00338

```
@prefix rr: <http:www.w3,org/ns/r2rml#>.
@prefix ex: <http://example.com/>.
<TriplesMap>
    rr:logicalTable [rr:sqlQuery """SELECT DISTINCT DrugBankID, Bioavailability, CasNumber, DrugName, DrugBankID FROM Drug """];
    rr:subjectMap [
        rr:template "http://example.com/Drug/{DrugBankID}";
        rr: class ex:Drug;];

    rr:predicateObjectMap [
        rr:predicate ex:biovailability ;
        rr:objectMap [rr:column "Bioavailability"; rr:datatype xsd:float];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:casNumber ;
        rr:objectMap [rr:column "CasNumber"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:drugName ;
        rr:objectMap [rr:column "DrugName"; rr:language "en-us"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:DrugBank;
        rr:objectMap [rr:column "DrugBankID"];
    ].
```

Drug

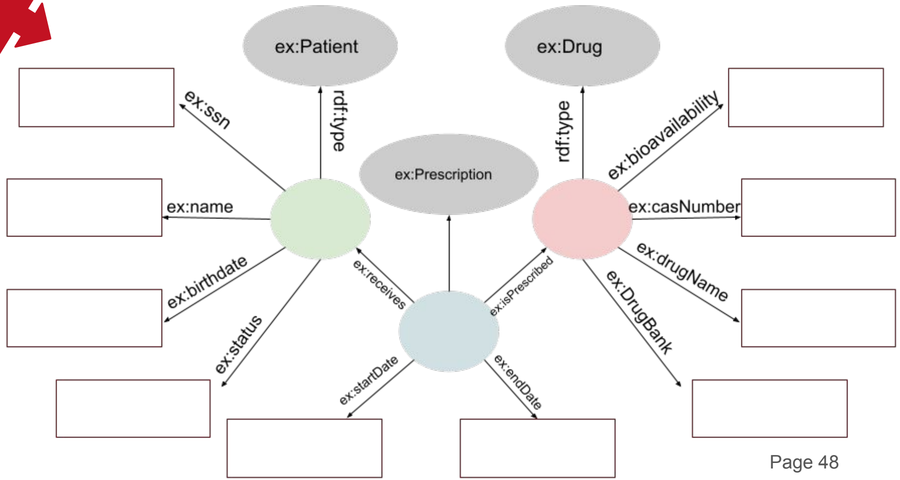
DrugName	Bioavailability	CasNumber	DrugBankID
Vinorelbine	43.000000	71486-22-1	DB00361
Cisplatine	100.000000	15663-27-1	DB00515
Omeprazole	35.000000	73590-58-6	DB00338

Patient

SSN	Name	Birthdate	Status
551	John Smith	20.12.1978	Alive with disease
552	Peter Lange	19.01.2010	Dead
553	Luis Perez	14.01.1959	Heatlhly

Treatment

DrugID	PatientID	StartDate	EndData
DB00361	551	20.01.2021	31.03.2021
DB00515	551	20.01.2021	31.03.2021
DB00338	551	20.01.2021	31.03.2021



Triples Map defining ex:Drug

```
@prefix rr: <http:www.w3,org/ns/r2rml#>.
@prefix ex: <http://example.com/>.
<TriplesMap1>
    rr:logicalTable [rr:tableName "Drug"];
    rr:subjectMap [
        rr:template "http://example.com/Drug/{DrugBankID}";
        rr: class ex:Drug;];

    rr:predicateObjectMap [
        rr:predicate ex:biovailability ;
        rr:objectMap [rr:column "Bioavailability"; rr:datatype xsd:float];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:casNumber ;
        rr:objectMap [rr:column "CasNumber"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:drugName ;
        rr:objectMap [rr:column "DrugName"; rr:language "en-us"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:DrugBank;
        rr:objectMap [rr:column "DrugBankID"];
    ].
```

Triples Map defining ex:Patient

```
<TriplesMap2>
    rr:logicalTable [rr:tableName "Patient"];
    rr:subjectMap [
        rr:template "http://example.com/Patient/{SSN}";
        rr: class ex:Patient;];

    rr:predicateObjectMap [
        rr:predicate ex:ssn ;
        rr:objectMap [rr:column "SSN"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:name ;
        rr:objectMap [rr:column "Name"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:birthdate ;
        rr:objectMap [rr:column "Birthdate"; rr:datatype xsd:date];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:status;
        rr:objectMap [rr:column "Status"; rr:language "en-us"];
    ].
```

Triples Map defining the ex:Prescription Relationship

```
<TriplesMap3>
    rr:logicalTable [rr:tableName "Treatment"];
    rr:subjectMap [
        rr:template "http://example.com/Prescription/{DrugID}_{PatientID}";
        rr: class ex:Prescription;];

    rr:predicateObjectMap [
        rr:predicate ex:receives ;
        rr:objectMap [rr:parentTriplesMap <TriplesMap2>;
        rr:joinCondition [
           rr:child "PatientID" ;
           rr:parent "SSN" ;];];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:isPrescribed ;
        rr:objectMap [rr:parentTriplesMap <TriplesMap1>;
        rr:joinCondition [
           rr:child "DrugID" ;
           rr:parent "DrugBankID" ;];];];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:startDate ;
        rr:objectMap [rr:column "StartDate"; rr:datatype xsd:date];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:startDate;
        rr:objectMap [rr:column "EndDate"; rr:datatype xsd:date];
    ].
```

Join between Triples Maps

```
<TriplesMap2>
    rr:logicalTable [rr:tableName "Patient"];
    rr:subjectMap [
        rr:template "http://example.com/Patient/{SSN}";
        rr: class ex:Patient;];

    rr:predicateObjectMap [
        rr:predicate ex:ssn ;
        rr:objectMap [rr:column "SSN"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:name ;
        rr:objectMap [rr:column "Name"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:birthdate ;
        rr:objectMap [rr:column "Birthdate"; rr:datatype xsd:date];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:status;
        rr:objectMap [rr:column "Status"; rr:language "en-us"];
    ].
```

```
<TriplesMap3>
    rr:logicalTable [rr:tableName "Treatment"];
    rr:subjectMap [
        rr:template "http://example.com/Prescription/{DrugID}_{PatientID}";
        rr: class ex:Prescription;];

    rr:predicateObjectMap [
        rr:predicate ex:receives ;
        rr:objectMap [rr:parentTriplesMap <TriplesMap2>;
        rr:joinCondition [
           rr:child "PatientID" ;
           rr:parent  "SSN"  ;];];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:isPrescribed ;
        rr:objectMap [rr:parentTriplesMap <TriplesMap1>;
        rr:joinCondition [
           rr:child "DrugID" ;
           rr:parent  "DrugBankID"  ;];];];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:startDate ;
        rr:objectMap [rr:column "StartDate"; rr:datatype xsd:date];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:startDate;
        rr:objectMap [rr:column "EndDate"; rr:datatype xsd:date];
    ].
```

```
@prefix rr: <http:www.w3,org/ns/r2rml#>.
@prefix ex: <http://example.com/>.
<TriplesMap1>
    rr:logicalTable [rr:tableName "Drug"];
    rr:subjectMap [
        rr:template "http://example.com/Drug/{DrugBankID}";
        rr: class ex:Drug;];

    rr:predicateObjectMap [
        rr:predicate ex:bioavailability ;
        rr:objectMap [rr:column "Bioavailability"; rr:datatype xsd:float];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:casNumber ;
        rr:objectMap [rr:column "CasNumber"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:drugName ;
        rr:objectMap [rr:column "DrugName"; rr:language "en-us"];
    ];

    rr:predicateObjectMap [
        rr:predicate ex:DrugBank;
        rr:objectMap [rr:column "DrugBankID"];
    ].
```

R2RML Triples Maps- Abstract Description

Given **DIS**=<**O**,**S**,**M**>, mapping rules in M are defined as safe horn clauses

$$body(\overline{X}) :- head(\overline{Y})$$

$body(\overline{X})$: conjunctive query over the alphabet of the data sources S with variables in $\overline{X}$

$head(\overline{Y})$ conjunction of predicates representing classes and properties in O with variables in $\overline{Y}$

$\overline{Y}$ subset of $\overline{X}$

Types of Mapping Rules

Concept Mapping Assertions: a conjunctive query over the predicate symbols of data sources to create the instances of a class C in the ontology O; *f(.)* is a function symbol

$$body(\overline{X}) : -C(f(y))$$

```
rr:logicalTable [rr:tableName "Treatment"];
rr:subjectMap [
    rr:template "http://example.com/Prescription/{DrugID}_{PatientID}";
    rr: class ex:Prescription;];
```

body(.)

f(y)

C(.)

Types of Mapping Rules

Role Mapping Assertions: a conjunctive query over the predicate symbols of data sources to create the arguments of *P(.,.)* is a predicate in the ontology O *f1(.)* and f2(.) are function symbols

$$body(\overline{X}) : -P(f_1(y_2), f_2(y_2))$$

```
<TriplesMap3>
    rr:logicalTable [rr:tableName "Treatment"];
    rr:subjectMap [
        rr:template "http://example.com/Prescription/{DrugID}_{PatientID}";
        rr: class ex:Prescription;];

    rr:predicateObjectMap [
        rr:predicate ex:receives ;
        rr:objectMap [rr:template "http://example.com/Patient/{PatientID}";
                      rr:termType rr:IRI];
    ];
```

body(.) *f1(y)* *P(.,.)* *f2(y)*

Types of Mapping Rules

Role Mapping Assertions: a conjunctive query over the predicate symbols of data sources to create the arguments of *P(.,.)* is a predicate in the ontology O *f1(.)* and f2(.) are function symbols

$$body(\overline{X}) :- P(f_1(y_2), f_2(y_2))$$

body(.) *f1(y)*

```
<TriplesMap3>
    rr:logicalTable [rr:tableName "Treatment"];
    rr:subjectMap [
        rr:template "http://example.com/Prescription/{DrugID}_{PatientID}";
        rr: class ex:Prescription;];

    rr:predicateObjectMap [
        rr:predicate ex:receives ;
        rr:objectMap [rr:parentTriplesMap <TriplesMap2>
        rr:joinCondition [
           rr:child "PatientID" ;
           rr:parent "SSN" ;];];
    ];
```

f2(y)

```
<TriplesMap2>
    rr:logicalTable [rr:tableName "Pati   ];
    rr:subjectMap [
       rr:template "http://example.com/Patient/{SSN}";
       rr: class ex:Patient;];
```

Types of Mapping Rules

Attribute Mapping Assertions: a conjunctive query over the predicate symbols of data source to create the arguments of an attribute *A(.,.)* in the ontology O; *f(.)* is a function symbol

$$body(\overline{X}) :- A(f(y_1), y_2)$$

```
@prefix rr: <http:www.w3,org/ns/r2rml#>.
@prefix ex: <http://example.com/>.
<TriplesMap>
    rr:logicalTable [rr:sqlQuery """SELECT DISTINCT DrugBankID, Bioavailability, CasNumber, DrugName, DrugBankID FROM Drug """];
    rr:subjectMap [
        rr:template "http://example.com/Drug/{DrugBankID}";
        rr: class ex:Drug;];

    rr:predicateObjectMap [
        rr:predicate ex:biovailability ;
        rr:objectMap [rr:column "Bioavailability"; rr:datatype xsd:float];
    ];
```

body(.)

f(y)

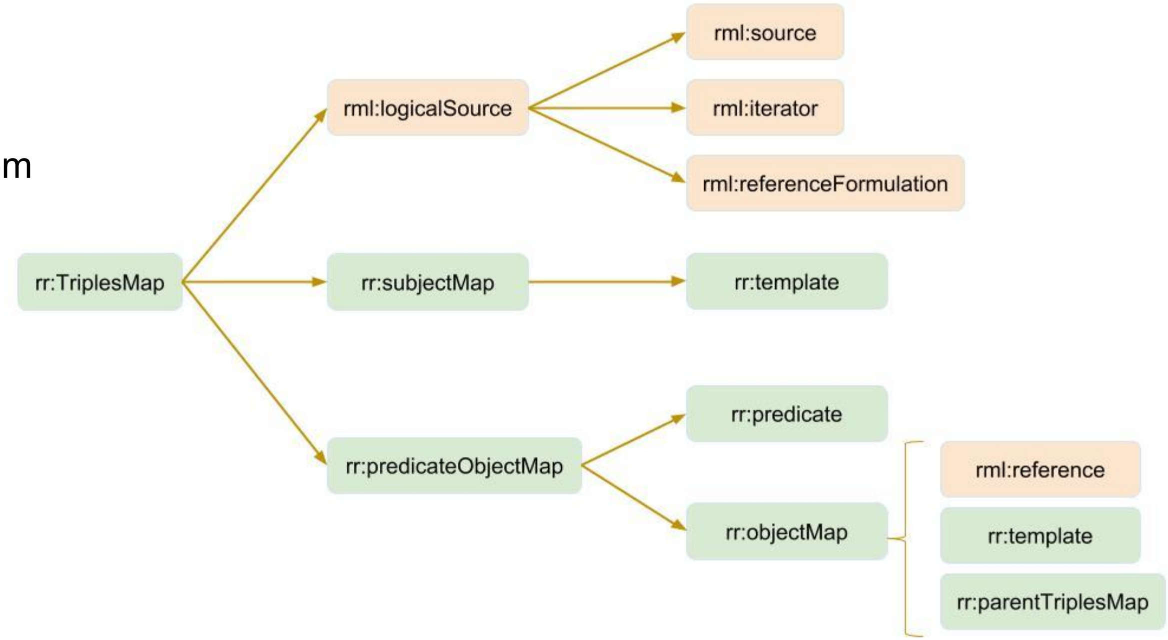
y2

A(.,.)

RDF Mapping Language

Mapping language defined on top of R2RML

- Enables the collection of data from data sources in various formats
  - XML, JSON, CSV, RDB



R2RML versus RML

R2RML	RML
Logical Table - only relational database (**rr:logicalTable**)	Logical Source - CSV, XML, JSON, HTML (**rml:logicalSource**)
Table Name (**rr:tableName**)	URI pointing to the source (**rml:source**) it can be RDB, JSON, XML, or CSV
Relational Table Column (**rr:column**)	Reference (**rml:reference**)
SQL query (**rr:sqlQuery**)	Reference Formulation (**rml:referenceFormulation**) • type of the source of the input data file, e.g. CSV, JSONPath, XPath.
Iteration per row in table	Defined iterator (**rml:iterator**)

Examples- Logical Data Sources

dataSource1.csv

DrugName	DrugBankID	DBpediaURL	UMLS CUI	UMLS Label
Vinorelbine	DB00361	http://dbpedia.org/resource/Vinorelbine	C0078257	Vinorelbine
Nivolumab	DB09035	http://dbpedia.org/resource/Nivolumab	C3657270	Nivolumab
Cisplatin	DB00515	http://dbpedia.org/resource/Cisplatin	C0008838	Cisplatin
Omeprazole	DB00338	http://dbpedia.org/resource/Omeprazole	C0028978	Omeprazole

dataSource2.csv

PatientID	PatientName	PrescribedDrug	StartDateTreatment	EndDateTreatment	Doses
5553	John Smith	Vinorelbine	02.12.2020	02.02.2021	3mg
5553	John Smith	Cisplatin	02.12.2020	02.02.2021	4mg
5554	Markus Hass	Omeprazole	04.10.2020	02.12.2020	250mg
5554	Markus Hass	Nivolumab	04.10.2020	02.12.2020	4mg

Example- RMl Mapping Rules to define class Drug

```
@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 schema: <http://schema.org/> .
@base <http://tib.de/ontorio/mapping> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix iasis: <http://project-iasis.eu/vocab/> .


<TriplesMap1>
  rml:logicalSource [
      rml:source "dataSource1.csv";
      rml:referenceFormulation ql:CSV
    ];

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

  rr:predicateObjectMap [
    rr:predicate iasis:drugLabel;
    rr:objectMap [
      rr:reference "DrugName";
      rr:datatype xsd:string
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate iasis:drugBankID;
    rr:objectMap [
      rr:reference "DrugBankID";
      rr:datatype xsd:string
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate owl:sameAs;
    rr:objectMap [
      rr:template "{DBpediaURL}"
    ]
  ].
```

Example- RML Mapping Rules to define class Patient

```
<TriplesMap2>
  rml:logicalSource [
      rml:source "dataSource2.csv";
      rml:referenceFormulation ql:CSV
    ];

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

  rr:predicateObjectMap [
    rr:predicate iasis:PatientName;
    rr:objectMap [
      rr:reference "PatientName";
      rr:datatype xsd:string
    ]
  ].
```

```
<TriplesMap3>
rml:logicalSource [
    rml:source "dataSource2.csv";
    rml:referenceFormulation ql:CSV
  ];

rr:subjectMap [
  rr:template "http://project-iasis.eu/PrescribedTreatment/{PatientID}_{PrescribedDrug}_{StartDateTreatment}_{EndDateTreatment}";
  rr:class iasis:PrescribedTreatment
];

rr:predicateObjectMap [
  rr:predicate iasis:patientID;
  rr:objectMap [
       rr:parentTriplesMap <TriplesMap2> ]
];

rr:predicateObjectMap [
  rr:predicate iasis:prescribedDrug;
  rr:objectMap [
    rr:parentTriplesMap <TriplesMap1>
    rr:joinCondition [ rr:child "PrescribedDrug"; rr:parent "DrugName"];

  ]
];

rr:predicateObjectMap [
  rr:predicate iasis:startDateTreatment;
  rr:objectMap [
    rr:reference "StartDateTreatment";
    rr:datatype xsd:date
  ]
];

rr:predicateObjectMap [
  rr:predicate iasis:endDateTreatment;
  rr:objectMap [
    rr:reference "EndDateTreatment";
    rr:datatype xsd:date
  ]
];
rr:predicateObjectMap [
  rr:predicate iasis:doses;
  rr:objectMap [
    rr:reference "Doses";
    rr:datatype xsd:string
  ]
].
```

Example- RML Triple Map over an RDB Logical Source

```
@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 schema: <http://schema.org/> .
@base <http://tib.de/ontorio/mapping> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix iasis: <http://project-iasis.eu/vocab/> .


<#Class1>
  rml:logicalSource [
    rml:source <RDB_source>;
    rr:sqlVersion rr:SQL2008;
    rml:query """Select DISTINCT DrugName, DrugBankID, DBpediaURL FROM Drugs""";
  ];
  rr:subjectMap [
    rr:template "http://project-iasis.eu/Drug/{DrugBankID}";
    rr:class iasis:Drug
  ];

  rr:predicateObjectMap [
    rr:predicate iasis:drugLabel;
    rr:objectMap [
      rr:reference "DrugName"
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate iasis:drugBankID;
    rr:objectMap [
      rr:reference "DrugBankID"
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate owl:sameAs;
    rr:objectMap [
      rr:template "{DBpediaURL}"
    ]
  ].

<RDB_source> a d2rq:Database;
d2rq:jdbcDSN "ExampleBIOMEDAS";
d2rq:jdbcDriver "com.mysql.cj.jdbc.Driver";
d2rq:username "";
d2rq:password "" .
```

rml:logicalSource

rr:subjectMap

rr:predicateObjectMap

Definition of the access to the database

Tracing DIS using Declarative Mapping Rules

Classes' Definition

```
SELECT DISTINCT  ?class ?typeDefinition ?source   WHERE
          {?triplesmap a <http://www.w3.org/ns/r2rml#TriplesMap> .
            ?triplesmap   <http://semweb.mmlab.be/ns/rml#logicalSource> ?o .
            ?o  ?typeDefinition                                          ?source .
            ?triplesmap  <http://www.w3.org/ns/r2rml#subjectMap>         ?o2 .
            ?o2          <http://www.w3.org/ns/r2rml#class>              ?class . }
```

Predicates' Definition

```
SELECT DISTINCT  ?class ?property ?definition ?objectValue WHERE
      {?triplesmap a <http://www.w3.org/ns/r2rml#TriplesMap> .
        ?triplesmap <http://semweb.mmlab.be/ns/rml#logicalSource>    ?o .
        ?o          ?typeDefinition                                  ?source .
        ?triplesmap <http://www.w3.org/ns/r2rml#subjectMap>          ?o2 .
        ?o2         <http://www.w3.org/ns/r2rml#class>               ?class .
        ?triplesmap <http://www.w3.org/ns/r2rml#predicateObjectMap>  ?o4 .
        ?o4         <http://www.w3.org/ns/r2rml#predicate>           ?property.
        ?o4         <http://www.w3.org/ns/r2rml#objectMap>           ?o6 .
        ?o6         ?definition   ?objectValue }
```

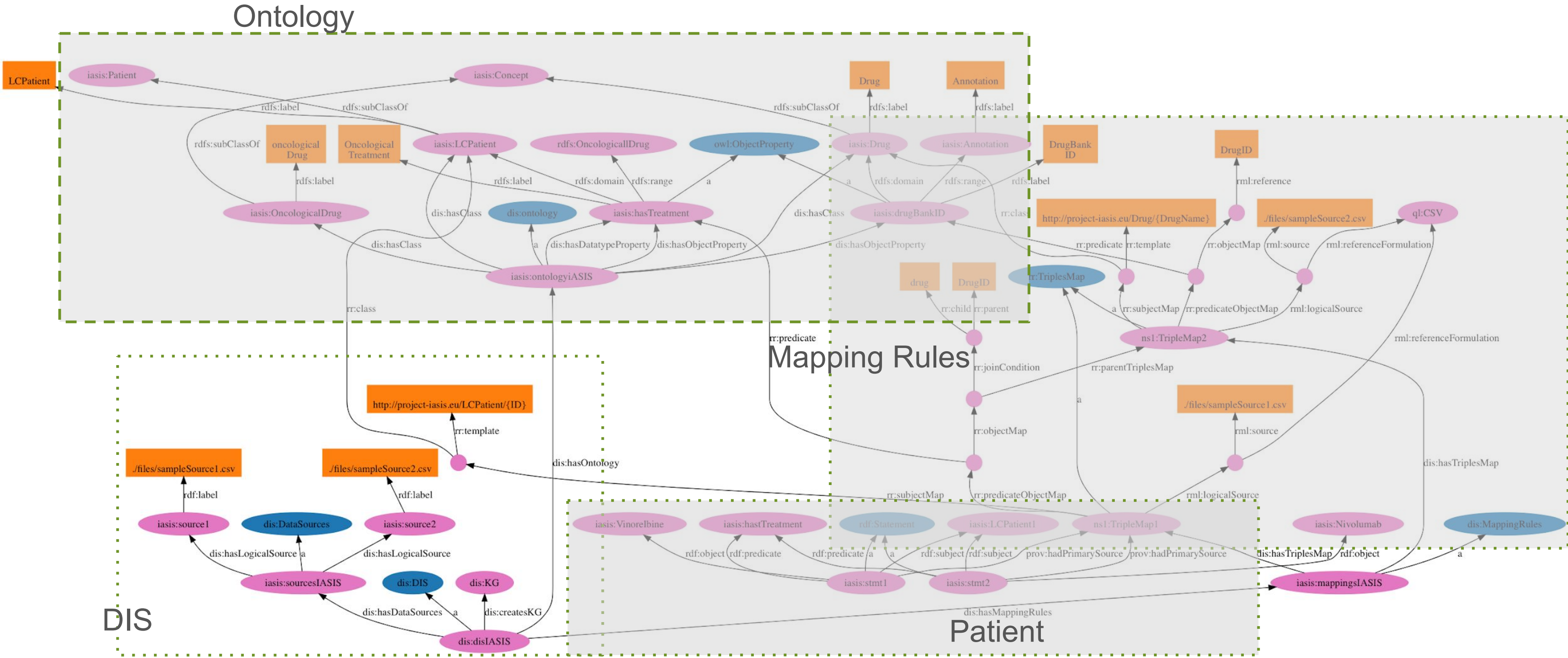
Number Mappings Per Class

```
SELECT ?class count(DISTINCT ?triplesmap) as ?numberMappings  WHERE
          {?triplesmap a <http://www.w3.org/ns/r2rml#TriplesMap> .
            ?triplesmap   <http://semweb.mmlab.be/ns/rml#logicalSource> ?o .
            ?o  ?typeDefinition                                        ?source .
            ?triplesmap  <http://www.w3.org/ns/r2rml#subjectMap>         ?o2 .
            ?o2          <http://www.w3.org/ns/r2rml#class>              ?class . }

GROUP BY ?class ORDER BY DESC(?numberMappings)
```

Classes used in Mapping Rules but not in Ontology

```
SELECT DISTINCT ?class   WHERE
          {?triplesmap a <http://www.w3.org/ns/r2rml#TriplesMap> .
            ?triplesmap   <http://semweb.mmlab.be/ns/rml#logicalSource> ?o .
            ?o  ?typeDefinition                                          ?source .
            ?triplesmap  <http://www.w3.org/ns/r2rml#subjectMap>         ?o2 .
            ?o2          <http://www.w3.org/ns/r2rml#class>              ?class .
          FILTER ( !EXISTS { ?class a owl:Class })}
```

BENEFITS OF A DECLARATIVE KG CREATION



Evaluating R2RML and RML Mapping Rules

Various Parameters Impact Execution Time

Engine	Execution time (secs.)	Number of results
	Two POM	
RMLMapper	0.92	2,000
SDM-RDFizer	1.72	2,000
	Five POM	
RMLMapper	1.84	5,000
SDM-RDFizer	1.85	5,000
	Ten POM	
RMLMapper	3.36	10,000
SDM-RDFizer	1.98	10,000

Number of POMs (PropertyObjectMap)

Engine	Execution time (secs.)	Number of results
	High Selectivity	
RMLMapper	38.6	2,100
SDM-RDFizer	2.16	2,100
	Medium Selectivity	
RMLMapper	40.43	23,000
SDM-RDFizer	2.20	23,000
	Low Selectivity	
RMLMapper	46.06	30,000
SDM-RDFizer	2.19	30,000

Join Selectivity

Engine	Execution time (secs.)	Number of results
	Low percentage of duplicates	
RMLMapper	37.94	20,027
SDM-RDFizer	2.01	20,027
	Medium percentage of duplicates	
RMLMapper	39.201	20,105
SDM-RDFizer	1.87	20,105
	High percentage of duplicates	
RMLMapper	40.81	20,263
SDM-RDFizer	1.89	20,263

Percentage of Duplicates

Engines are not equality impacted by DIS configurations

Engine	Execution time (secs.)	Number of results
	1-1	
RMLMapper	42.86	25,000
SDM-RDFizer	2.19	25,000
	1-N	
RMLMapper	43.34	22,490
SDM-RDFizer	2.19	22,490
	N-1	
RMLMapper	43.26	22,490
SDM-RDFizer	2.15	22,490
	N-M	
RMLMapper	78.64	25,200
SDM-RDFizer	2.33	25,200

Type of Joins

Engine	Execution time (secs.)	Number of results
	Horizontal Partitioning without Replication	
RMLMapper	1,904.31	310,000
SDM-RDFizer	4.84	310,000
	Vertical Partitioning without Replication	
RMLMapper	2,067.77	310,000
SDM-RDFizer	4.73	310,000
	Horizontal Partitioning with Replication	
RMLMapper	2,276.98	310,000
SDM-RDFizer	5.86	310,000
	Vertical Partitioning with Replication	
RMLMapper	2,024.66	310,000
SDM-RDFizer	4.98	310,000

Data Partition

David Chaves-Fraga, Kemele M. Endris, Enrique Iglesias, Óscar Corcho, Maria-Esther Vidal: **What Are the Parameters that Affect the Construction of a Knowledge Graph?** ODBASE 2019

RML Operators

```
<TriplesMap1>
 rml:logicalSource [ rml:source "dataSource1" ;
                     rml:referenceFormulation ql:CSV ];
 rr:subjectMap [
   rr:template "http://example.org/Gene/{Gene name}";
   rr:class ex:Gene];

 rr:predicateObjectMap [
   rr:predicate ex:geneLabel;
    rr:objectMap [ rml:reference "Gene name" ] ];

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

<TriplesMap2>
 rml:logicalSource [ rml:source "dataSource1" ;
                     rml:referenceFormulation ql:CSV];
 rr:subjectMap [
   rr:template "http://example.org/Sample/{ID_sample}";
   rr:class ex:Sample] ;

 rr:predicateObjectMap [
   rr:predicate ex:sample_isTakenFrom_tumor;
   rr:objectMap [
     rr:parentTriplesMap <TriplesMap3>;
     rr:joinCondition [ rr:child "ID_sample";
                        rr:parent "ID_sample" ;];].

<TriplesMap3>
 rml:logicalSource [ rml:source "dataSource2";
                     rml:referenceFormulation ql:JSONPath
                     rml:iterator "$.[*]"];
 rr:subjectMap [
   rr:template "http://example.org/Tumor/{ID_tumor}";
   rr:class ex:Tumor ] .
```

Simple Object Map (SOM) evaluates predicate object map in **triples maps**

Object Reference Map (ORM) implements a **reference** between **two triples maps**

Object Join Map (OJM) implements a **join condition** between two triples maps

Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, Maria-Esther Vidal. **SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs.** ACM CIKM 2020.

Logical Operators in Triples Maps

Simple Object Map (SOM): Given a source **S**, a property **p**, and two attributes **A** and **B** of S, **SOM(S,p,A,B)** generates RDF triples **(a, p, b),** by projecting the values of **A** and **B** from **S**. **SOM** corresponds to the **PROJECT** operator π.

$$SOM(S,p,A,B)=\{(a,p,b) \mid (a,b) \in \pi_{A,B}(S)\}$$

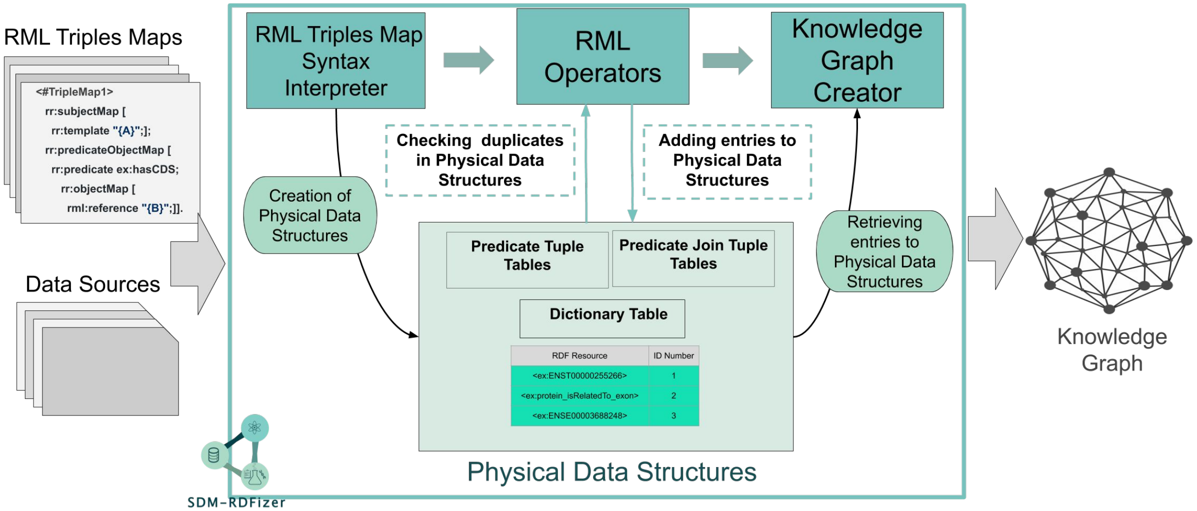
Object Reference Map (ORM): Given two sources **S1** and **S2,** a property **p,** and attributes **A and B from S1** and **S2**, respectively. **ORM(S1,S2,p,A,B)** generates RDF triples **(a,p,b)** by projecting the attributes **A** and **B** from the **natural join** of **S1** and **S2**.

$$ORM(S1,S2,p,A,B)=\{(a,p,b) \mid (a,b) \in \pi_{A,B}(S1 \bowtie S2)\}$$

Object Join Map (OJM): Given two sources **S1** and **S2,** a property **p,** and attributes **A and B from S1** and **S2**, respectively. Let $\boldsymbol{\sigma}$ be a join condition on attributes of **S1** and **S2**. **OJM(S1,S2,p,A,B,σ)** generates RDF triples **(a,p,b)** by projecting the attributes **A** and **B** from the **genaralized join** of **S1** and **S2** on $\boldsymbol{\sigma}$.

$$OJM(S1,S2,p,A,B,\sigma)=\{(a,p,b) \mid (a,b) \in \pi_{A,B}(S1 \bowtie_{\sigma} S2)\}$$

DECLARATIVE KNOWLEDGE GRAPH CREATION



Physical Data Structures avoid the generation of duplicated triples

- **Predicate Tuple Table (PTT):** for each predicate **p**, stores all the triples generated so far
- **Predicate Join Table (PJTT):** stores the subjects of the triples generated by a join.

SDM-RDFizer implements **three physical operators**:

- Simple Object Map (SOM)
- Object Reference Map (ORM)
- Object Join Map (OJM)

Iglesias et al. SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs. ACM CIKM 2020.
Jozashoori et al. FunMap: Efficient Execution of Functional Mappings for Scaled-Up Knowledge Graph Creation. ISWC 2020

Predicate Tuple Table (PTT)

Predicate Tuple Table (PTT)

- stores RDF triples for each predicate generated so far
- **Key** encoding **subject** and **object**

```
<TriplesMap1>
   rml:logicalSource [ rml:source "dataSource1" ];
    rr:subjectMap [
      rr:template "http://example.org/Gene/{Gene Name}";
      rr:class ex:Gene];
    rr:predicateObjectMap [
      rr:predicate ex:geneLabel;
      rr:objectMap [ rml:reference "Gene Name" ] ];
```

<http://example.org/Gene/PHF12_ET00000268756> <ex:geneLabel> “PHF12_ET00000268756”.
<http://example.org/Gene/ALDH3A1_ET00000395555> <ex:geneLabel> “ALDH3A1_ET00000395555”.

Key
encode(http://example.org/Gene/PHF12_ET00000268756, PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555, ALDH3A1_ET00000395555)

PTT ex:geneLabel

Predicate Join Tuple Table (PJTT)

```
<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template
"http://example.org/Sample/{ID_sample}";
     rr:class ex:Sample] ;
   rr:predicateObjectMap [
     rr:predicate ex:sample_isTakenFrom_tumor;
     rr:objectMap [
       rr:parentTriplesMap <TriplesMap3>;
       rr:joinCondition [ rr:child "ID_sample";
                          rr:parent "ID_sample" ;];].

 <TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2" ];
   rr:subjectMap [
     rr:template "http://example.org/Tumor/{ID_tumor}";
     rr:class ex:Tumor ] .
```

Object Join Map (OJM)

Predicate Join Tuple Table (PJTT)

- stores values generated during execution of a join condition between two RML triple maps.

Index Hash table to the Source S2 of the parentTM:

- **Key** encoding of each of **value(s)** of the **attributes** in the **join condition**
- **Value** set with the subject values in S2 associated with the values of the attributes in the hash key

Predicate Join Tuple Table (PJTT)

```
<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template
"http://example.org/Sample/{ID_sample}";
     rr:class ex:Sample] ;
   rr:predicateObjectMap [
     rr:predicate ex:sample_isTakenFrom_tumor;
     rr:objectMap [
       rr:parentTriplesMap <TriplesMap3>;
       rr:joinCondition [ rr:child "ID_sample";
                          rr:parent "ID_sample" ;];].

 <TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2" ];
   rr:subjectMap [
     rr:template "http://example.org/Tumor/{ID_tumor}";
     rr:class ex:Tumor ] .
```

dataSource1

Gene Name	ID_sample
ALDH3A1_ET00000395555	2193351
PHF12_ET00000268756	2193351
PHF12_ET00000268756	2196270

dataSource2

ID_sample	ID_tumor
2193351	1455465
2193351	2064548
2196270	2061629

Predicate Join Tuple Table (PJTT)

```
<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template
"http://example.org/Sample/{ID_sample}";
     rr:class ex:Sample] ;
   rr:predicateObjectMap [
     rr:predicate ex:sample_isTakenFrom_tumor;
     rr:objectMap [
       rr:parentTriplesMap <TriplesMap3>;
       rr:joinCondition [ rr:child "ID_sample";
                          rr:parent "ID_sample" ;];].

  <TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2" ];
   rr:subjectMap [
     rr:template "http://example.org/Tumor/{ID_tumor}";
     rr:class ex:Tumor ] .
```

dataSource2

ID_sample	ID_tumor
2193351	1455465
2193351	2064548
2196270	2061629

TripleMap2_ID_sample	
[2193351]	[1455465,2064548]
[2196270]	[2061629]

JPTT TripleMap2_ID_sample

Dictionary Table

Dictionary Table

- Encodes each RDF resource with an identification number

Hash table:

- **Key** RDF resource
- **Value** identification number in base 36

Key
encode(http://example.org/Gene/PHF12_ET00000268756_PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555_ALDH3A1_ET00000395555)

PTT ex:geneLabel

Dictionary Table

Key	Value
http://example.org/Gene/PHF12_ET00000268756	1
ex:geneLabel	2
“PHF12_ET00000268756”	3
http://example.org/Gene/ALDH3A1_ET00000395555	4
“ALDH3A1_ET00000395555”	5

Dictionary Table

Dictionary Table

- Encodes each RDF resource with an identification number

Hash table:

- **Key** RDF resource
- **Value** identification number in base 36

Key
encode(http://example.org/Gene/PHF12_ET00000268756_PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555_ALDH3A1_ET00000395555)

PTT ex:geneLabel

Dictionary Table

Key	Value
http://example.org/Gene/PHF12_ET00000268756	1
ex:geneLabel	2
"PHF12_ET00000268756"	3
http://example.org/Gene/ALDH3A1_ET00000395555	4
"ALDH3A1_ET00000395555"	5

Key
1_3
4_5

PTT 2

Physical Operators

TIB

Simple Object Map (SOM):

Triples Map **tm1** defines predicate **p** on logical source **S**
and **tm1** subjectMap is **f1(att1)**
and **tm1** objectMap for **p** is **f2(att2)**

For each row in S

a. Create an RDF triple **t=(f1(row.att1),p,f2(row.att1))**
b. If **encode(f1(row.att1),f2(row.att1))** does not belong to the PPT for **p**
   i. **Add encode(f1(row.att1),f2(row.att1))** to PPT for **p**
   ii. **Output (f1(row.att1),p,f2(row.att1))** to the KG

<http://example.org/Gene/ALDH3A1_ET00000395599> <ex:geneLabel> “ALDH3A1_ET00000395599”.

+

Key
encode(http://example.org/Gene/PHF12_ET00000268756, PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555, ALDH3A1_ET00000395555)

→

Key
encode(http://example.org/Gene/PHF12_ET00000268756, PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555, ALDH3A1_ET00000395555)
encode(http://example.org/Gene/ALDH3A1_ET00000395599, ALDH3A1_ET00000395599)

Physical Operators

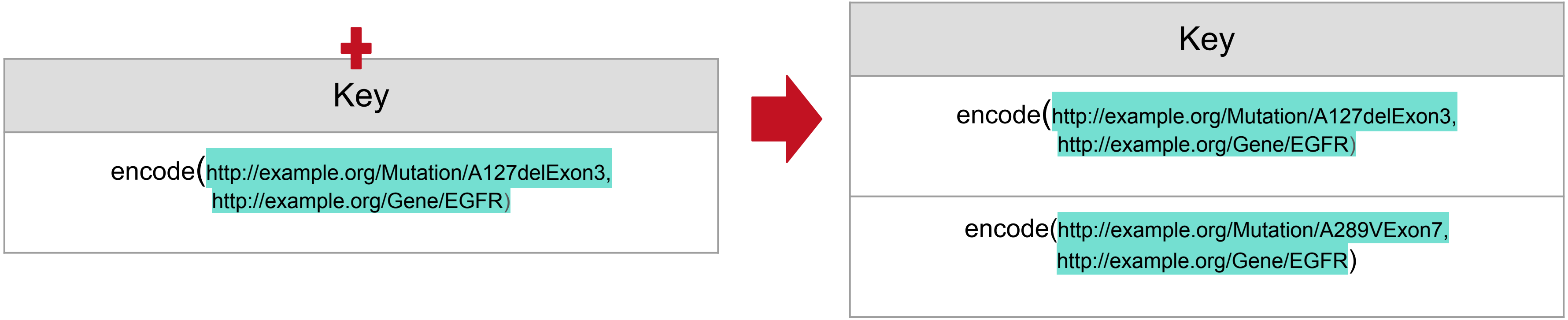
Object Reference Map (ORM):

Triples Map **tm1 refers** to Triples Map **tm2** to define predicate **p** on logical source **S**:
and subject of **tm1** is defined as **f1(att1)**
and subject of **tm2** is defined as **f2(att2)**

For each row in S

a. Create an RDF triple **t=(f1(row.att1),p,f2(row.att1))**
b. If **encode(f1(row.att1),f2(row.att1))** does not belong to the PPT for **p**
   i. **Add encode(f1(row.att1),f2(row.att1))** to PPT for **p**
   ii. **Output (f1(row.att1),p,f2(row.att1))** to the KG

<http://example.org/Mutation/A289VExon7> <ex:isMutation> <http://example.org/Gene/EGFR>



Physical Operators- OJM

```
<TriplesMap2>
  rml:logicalSource [ rml:source "dataSource1" ];
  rr:subjectMap [
   rr:template
"http://example.org/Sample/{ID_sample}";
   rr:class ex:Sample] ;
  rr:predicateObjectMap [
   rr:predicate ex:sample_isTakenFrom_tumor;
   rr:objectMap [
    rr:parentTriplesMap <TriplesMap3>;
    rr:joinCondition [ rr:child "ID_sample";
                 rr:parent "ID_sample" ;];].

 <TriplesMap3>
  rml:logicalSource [ rml:source "dataSource2" ];
  rr:subjectMap [
   rr:template "http://example.org/Tumor/{ID_tumor}";
   rr:class ex:Tumor ] .
```

dataSource1

Gene Name	ID_sample
ALDH3A1_ET00000395555	2193351
PHF12_ET00000268756	2193351
PHF12_ET00000268756	2196270

dataSource2

ID_sample	ID_tumor
2193351	1455465
2193351	2064548
2196270	2061629

For each row1 in dataSource1

If there is an entry in the attributes of the join condition

JPTT TripleMap2_ID_sample

TripleMap2_ID_sample	
[2193351]	[1455465,2064548]
[2196270]	[2061629]

Then extract the values associated with the entry and generate the corresponding entries in PPT

Key
encode(http://example.org/Sample/2193351,http://example.org/Tumor/1455465)
encode(http://example.org/Sample/2193351,http://example.org/Tumor/2064548)
encode(http://example.org/Sample/2196270,http://example.org/Tumor/2061629)

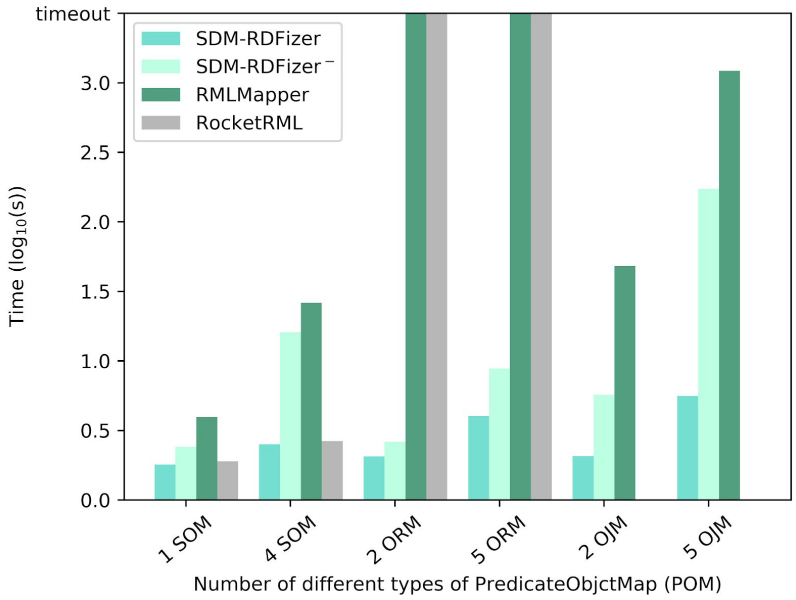
Empirical Evaluation

Data Sources:
COSMIC: Coding point mutation dataset. Raws were randomdly selected
Number of Rows: 10k, 100k, and 1M.
Duplicate Rates: 25%
Operators per Mappings: SOM (1-4), ORM (2-5), and OJM (2-5)

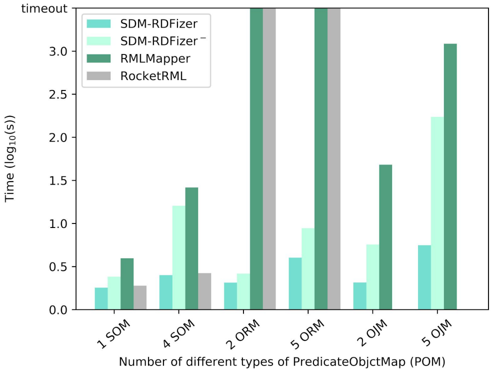
RML Engines:
SDM-RDFizer **v3.2**
RMLMapper **v4.7**
RocketRML **v1.7.0**

SDM-RDFizer$^-$ naive RML operators

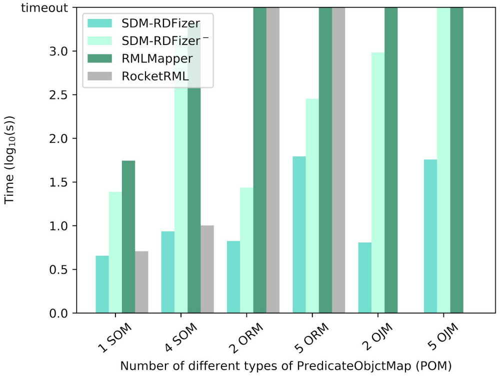
Execution time: Elapsed time in RDF KG creation (reported by the **time command** of the Linux operating system)
Timeout: Five hours



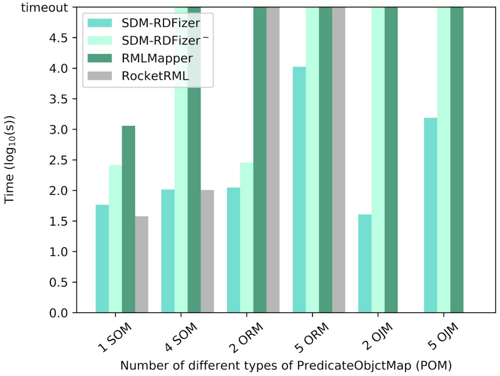
Empirical Evaluation



10K Rows



100K Rows



1M Rows

RocketRML v1.7.0 and **RMLMapper v4.7** timed out (2ORM, 5ORM, 2OJM, and 5OJM)
RMLMapper v4.7 failed executing 2 OJM and 5OJM
SDM-RDFizer physical operators speed up knowledge graph

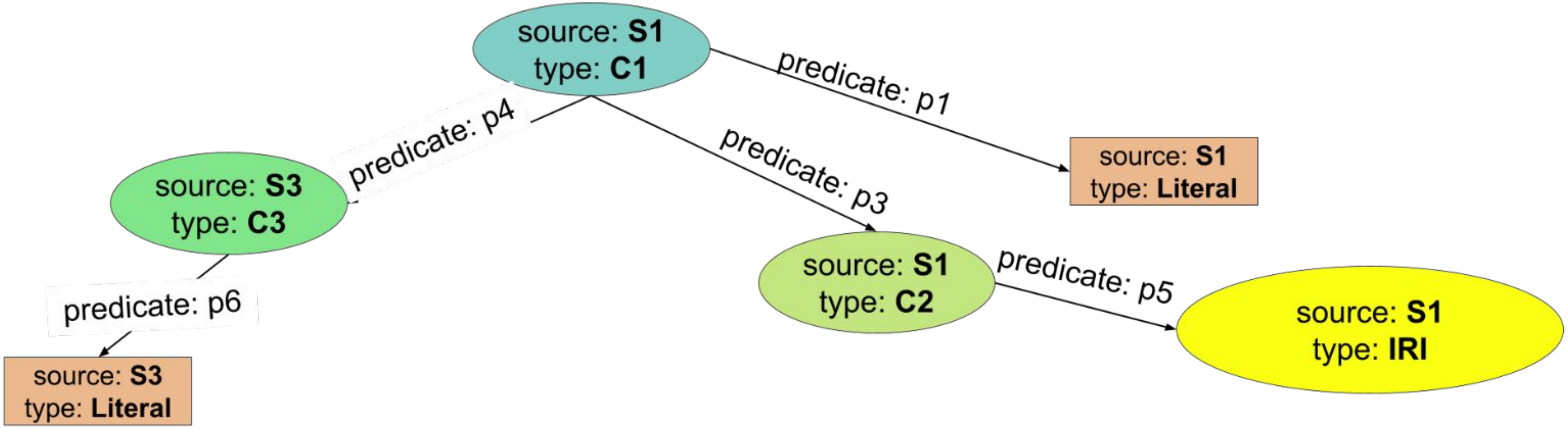
Similar performance is observed in testbeds with different duplication rates and size

How about planning the mapping rules?

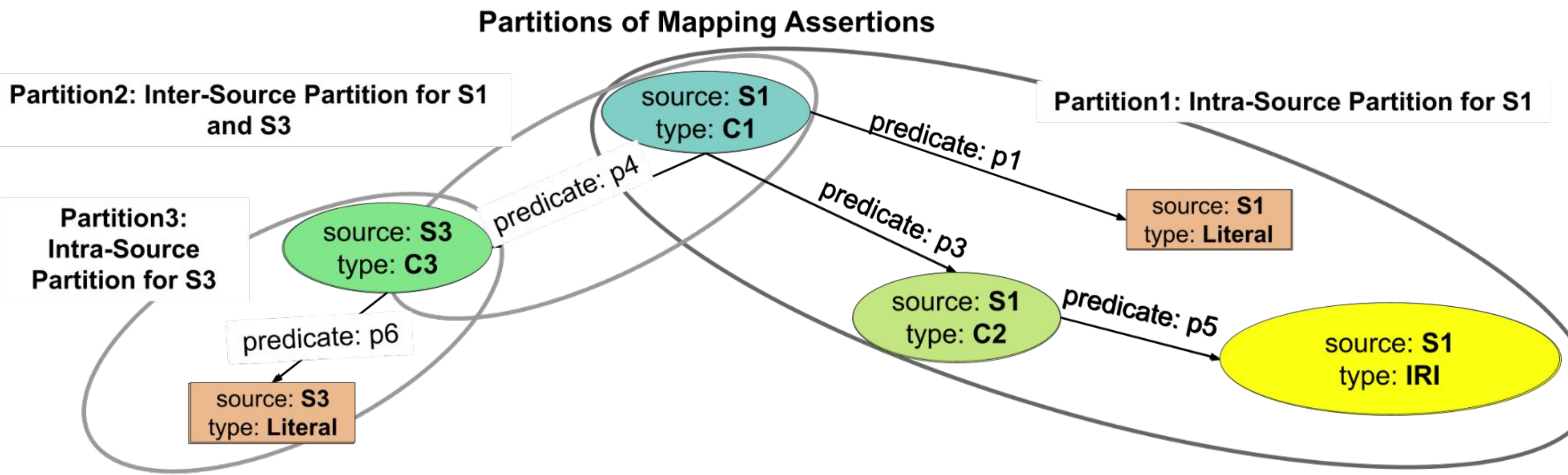
```
<TriplesMap1>
 rml:logicalSource [ rml:source "S1.csv"; ];

 rr:subjectMap [
   rr:template "http://www.example.com/C1{attribute1}"; ];
   rr:class ex:C1;

 rr:predicateObjectMap [
   rr:predicate ex:p1;
   rr:objectMap [ rml:reference "attributeX" ] ]; ].

 rr:predicateObjectMap
   rr:predicate ex:p3;
   rr:objectMap [ rr:parentTriplesMap <TriplesMap2> ] ].

 rr:predicateObjectMap [
   rr:predicate ex:p4;
   rr:objectMap [
   rr:parentTriplesMap <TriplesMap3>;
   rr:joinCondition [
       rr:child "attribute"
       rr:parent "DrugName" ]; ] ]

```

```
<TriplesMap2>
   rml:logicalSource [ rml:source "S1.csv";];
   rr:subjectMap [
     rr:template "http://www.example.com/C2{attribute2}";];
     rr:class ex:C2;

 rr:predicateObjectMap [
   rr:predicate ex:p5;
   rr:objectMap [ rr:template "https://dbpedia.org/{attributeY}";]].

<TriplesMap3>
   rml:logicalSource [ rml:source "S3.csv";];
   rr:subjectMap [
     rr:template "http://www.example.com/C3{attribute3}";];
     rr:class ex:C3;

   rr:predicateObjectMap [
   rr:predicate ex:p6;
   rr:objectMap [ rml:reference "attributeZ" ] ]; ].
```

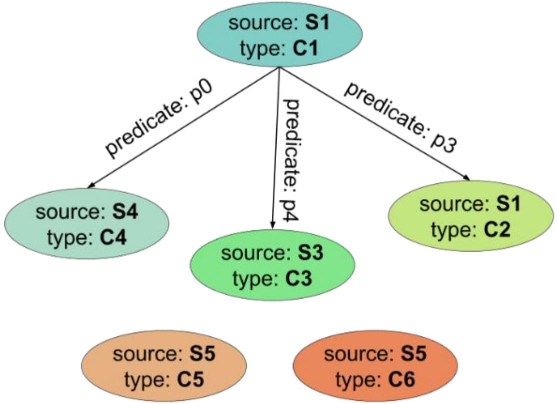
How about planning the mapping rules?



How about planning the mapping rules?

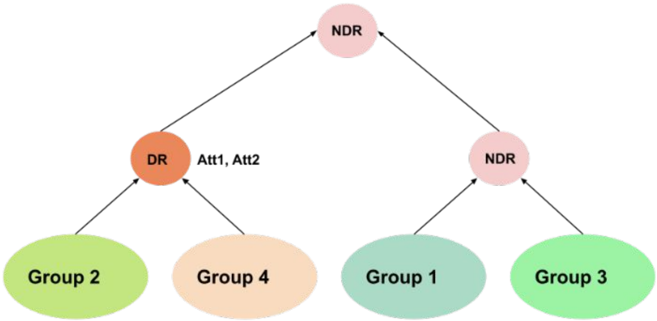


Impact of Planning the Execution of Mapping Rules



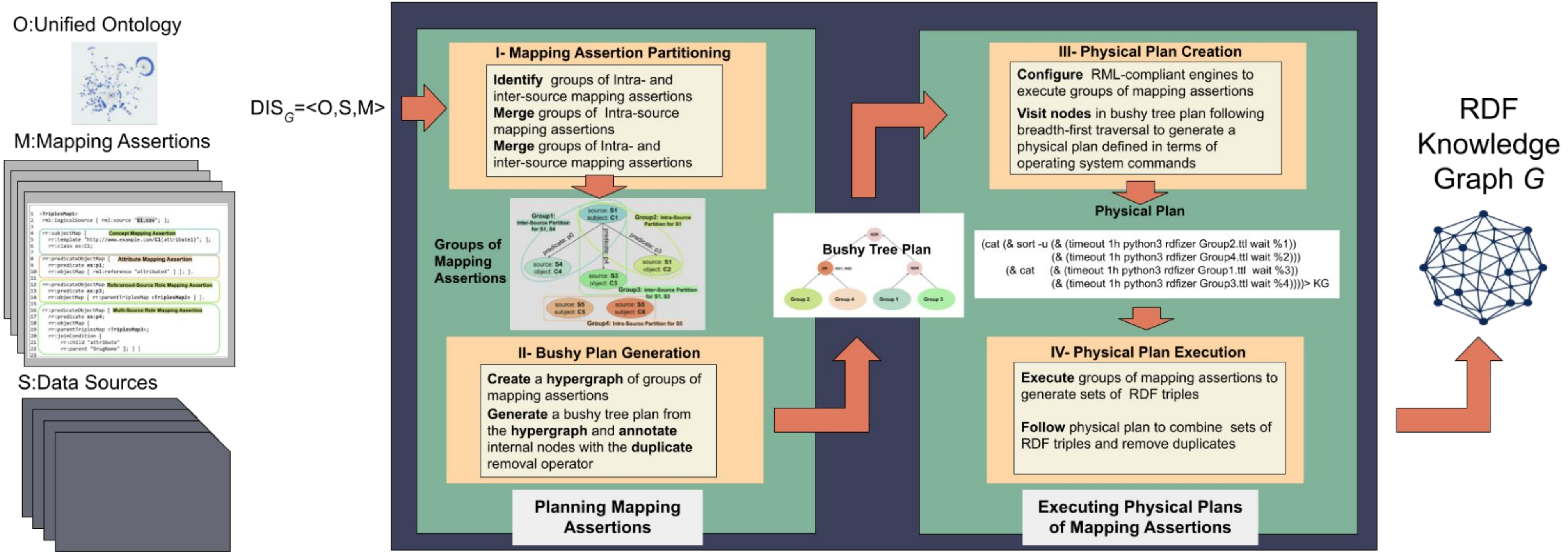
No Partitioning		
RMLMapper:	Five-hour timeout	(No results)
RocketRML:	Out of memory	(No results)
SDM-RDFizer:	441.18 sec	(100% results)
Morph-KGC:	42.32 sec	(100% results)

Options of Planning the Partitions of Mapping Rules

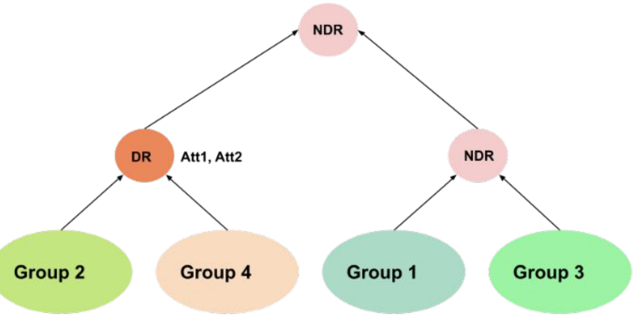


a) **Bushy Tree** where **Duplicate Removal** (**DR**) is pushed down

Pipeline for Planning and Executing Mapping Assertions



Iglesias, Jozashoori, Vidal, Scaling Up Knowledge Graph Creation to Large and Heterogeneous Data Sources. https://arxiv.org/abs/2201.09694



a) **Physical Plans** for a Bushy Tree with Duplicate Removal pushed down

Physical Plan Execution Time (Excluding Execution Time of Groups): 2.97 sec

SDM-RDFizer:

```
(cat (& sort -u (& (timeout 1h python3 rdfizer Group2.ttl wait %1))
                (& (timeout 1h python3 rdfizer Group4.ttl wait %2)))
     (& cat    (& (timeout 1h python3 rdfizer Group1.ttl  wait %3))
                (& (timeout 1h python3 rdfizer Group3.ttl wait %4))))> KG
```

Total Execution Time: 213.99 sec (100% results)

RMLMapper:

```
(cat (& sort -u (& (timeout 1h java rmlmapper Group2.ttl wait %1))
                 (& (timeout 1h java rmlmapper Group4.ttl wait %2)))
     (& cat     (& (timeout 1h java  rmlmapper Group1.ttl  wait %3))
                 (& (timeout 1h java rmlmapperGroup3.ttl wait %4))))> KG
```

Total Execution Time: Timeout (92.65% results)

RocketRML:

```
(cat (& sort -u (& (timeout 1h node rocketrml Group2.ttl wait %1))
                (& (timeout 1h node rocketrml Group4.ttl wait %2)))
     (& cat    (& (timeout 1h node  rocketrml Group1.ttl  wait %3))
                (& (timeout 1h node rocketrml Group3.ttl wait %4))))> KG
```

Total Execution Time: 136.79 sec (100% results)

Morph-KGC:

```
(cat (& sort -u (& (timeout 1h python3 morph_kgc Group2.ttl wait %1))
                (& (timeout 1h python3 morph_kgc  Group4.ttl wait %2)))
     (& cat     (& (timeout 1h python3 morph_kgc  Group1.ttl  wait %3))
                (& (timeout 1h python3 morph_kgc Group3.ttl wait %4))))> KG
```

Total Execution Time: 42.04 sec (100% results)

Empirical Evaluation

Benchmarks:

SDM-COSMIC* created by randomly selecting genomic mutation data in the COSMIC database**.

- Eight different logical data sources with various sizes including 10k, 100k, 1M, and 10M rows.
- Duplicate rates: 25% or 75%.
- Mapping assertion (MA) configurations:
  - **Conf7**: Four MAs (defining the same predicates) with four concepts and two multisource role MAs.
  - **Conf8**: Six MAs with six concepts and five multi-source role MAs. Five child MAs are referring to the same parent MA.
  - **Conf9**: Eight MAs with eight concepts and seven multi-source role MAs.

Engines:

- RMLMapper v4.12, Morph-KGC v1.4.1, SDM-RDFizer v3.6

Metrics:

- Execution Time

* https://figshare.com/articles/dataset/SDM-Genomic-Datasets/14838342/1

** https://cancer.sanger.ac.uk/cosmic

Impact of Planning Mapping Rules - Different Configurations

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 %

Impact of Planning Mapping Rules - Different Configurations

TIB

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%

Planning the execution of mapping rules:

- plays a crucial role in the KG creation process
- consumes time, in simple cases, it may generate overhead and negatively impact an engine behavior

Integrity Constraint Validation

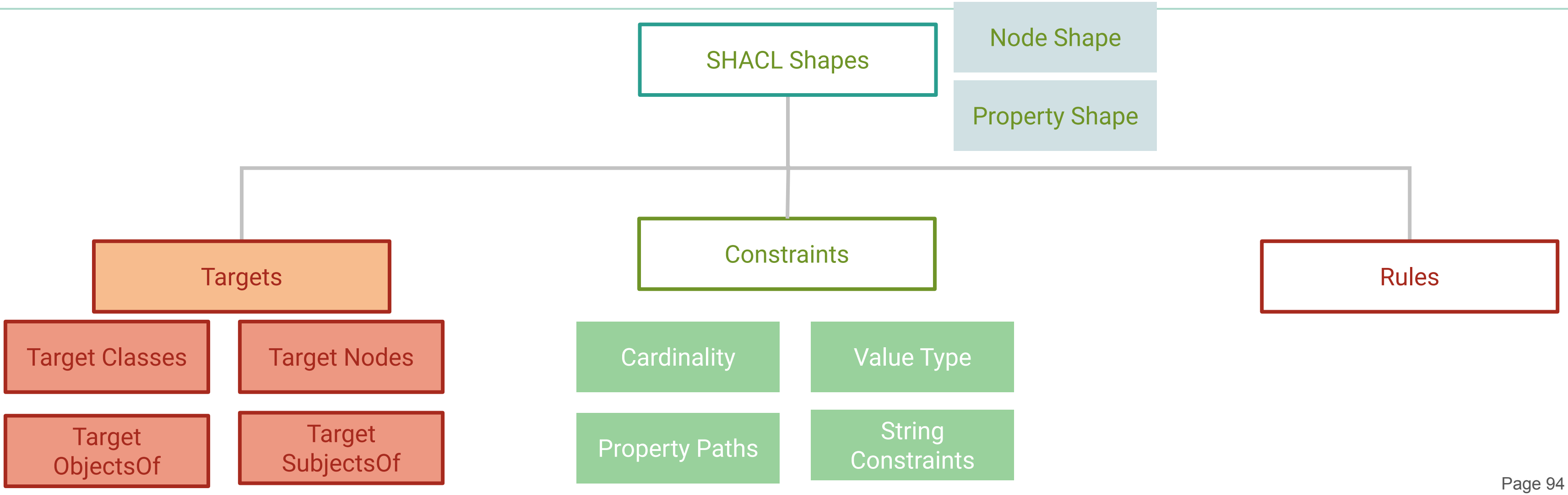
SHACL Language

The **SHA**pes **C**onstraint **L**anguage (SHACL):

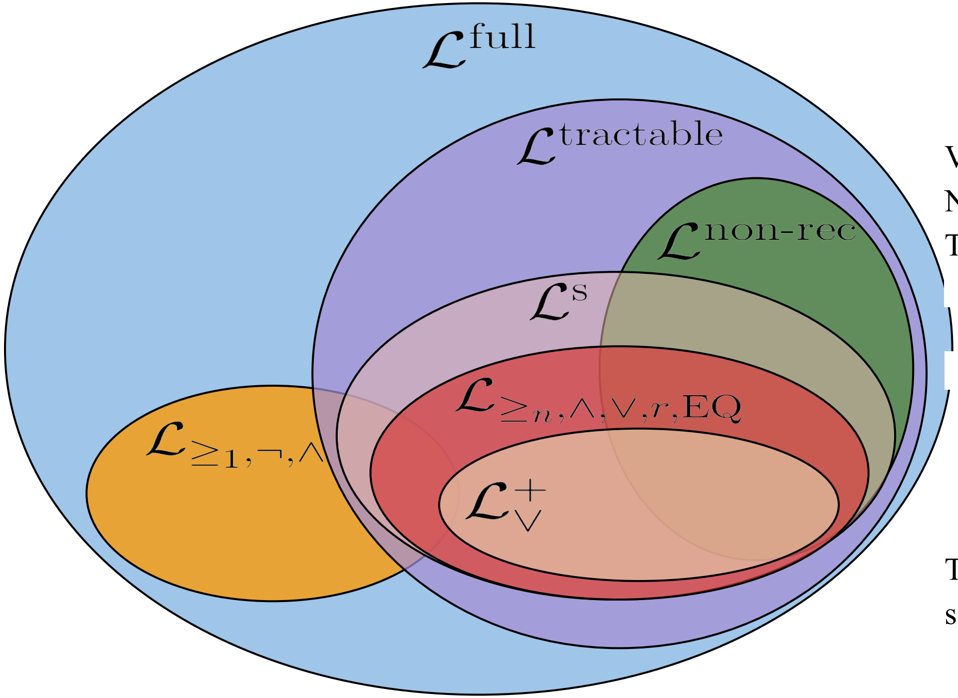
- W3C recommendation language for the declarative specification of integrity constraints over RDF KGs.

A SHACL shape:

- represents a set of constraints that apply over the same entities.
- can refer to another shape, two represent constraints between entities of two types.



SHACL Fragments



- Validating an RDF graph against SHACL constraints is NP-hard in the size of the graph [Corman et al. 2018]
- Tractable fragments of SHACL
  - $\mathcal{L}^{\text{non-rec}}$ only enables non-recursive shapes
  - $\mathcal{L}^{\text{S}}$ does not allow negations through recursive shapes
  - $\mathcal{L}^{+}_{\vee}$ does not allow negations but disjunction.

These fragments can be computed in polynomial time in the size of the result of the data required to validate the constraints

SHACL Example

The entities of class **LCPatient**:

- have exactly **one name**
- cannot have a treatment that includes **Nivolumab** and **Vinorelbine**

```
@prefix ex: <http://www.example.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .


ex:LCPatientShape rdf:type sh:NodeShape ;
                    sh:targetClass ex:LCPatient;
                    sh:property [
                    sh:path iasis:name ;
                    sh:minCount 1;
                    sh:maxCount 1 ;];
          sh:or [
                    sh:not [ sh:property
                        [sh:path ex:hasTreatment ;
                         sh:hasValue ex:Vinorlbine ];];
                    sh:not [ sh:property
                        [sh:path ex:hasTreatment ;
                         sh:hasValue ex:Nivolumab ;];];].
```

Transparency in Knowledge-driven Data Ecosystems-Example

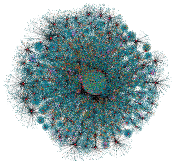
Lung Cancer Protocols:

Afatinib is a second generation Tyrosine Kinase Inhibitors (TKI) **not** recommended for non-small cell lung cancer patients with **Epidermal Growth Factor Receptor (EGFR)** mutation negative.

Lapatinib is a dual Tyrosine Kinase Inhibitors (TKI) for non-small cell lung cancer patients with **HER2** mutation positive or **EGFR** positive.

```
ex:NSLCProtocol1
        a sh:NodeShape ;
        sh:targetClass ex:NSLC-EGFR-negative ;
        sh:property [
                sh:path ex:hasOncologicalTreatment ;
                sh:hasValue dbpedia:Afatinib;
                sh:maxCount 0 ]

ex:NSLCProtocol2
        a sh:NodeShape ;
        sh:targetClass ex:NSLC-HER2-OR-EGFR-positive
        sh:property [
                sh:path ex:hasOncologicalTreatment ;
                sh:hasValue dbpedia:Lapatinib;
                sh:minCount 1 ]
```

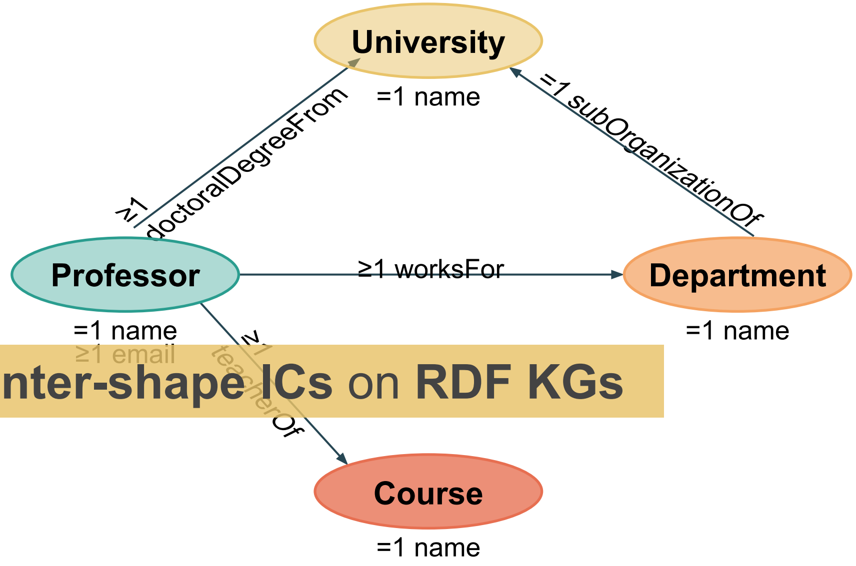
Motivating Example (1/2)



An RDF KG of a University System

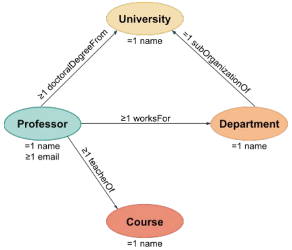
The data contains information about universities. Each university has to have one name. Professors are also present in the data. Each of them has exactly one name and at least one email address. Professors have at least one doctoral degree from a university. The knowledge graph also covers the departments of a university; they have exactly one name and are a sub-organization of a university. Professors work for at least one department. The university system also holds information about the courses taught. Each course has one name. Professors teach at least one course.

SHACL represents intra- and inter-shape ICs on RDF KGs

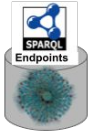


Graphical Representation of a SHACL Network

Impact of Shape Traversal on Validation Time



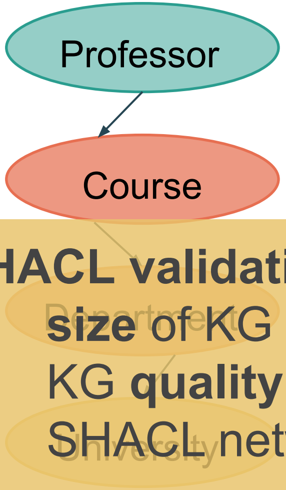
SHACL Network



Knowledge Graph
~1 million triples

Class	#entities	#valid
University	1000	8
Department	149	149
Professor	1267	7
Course	8126	8126

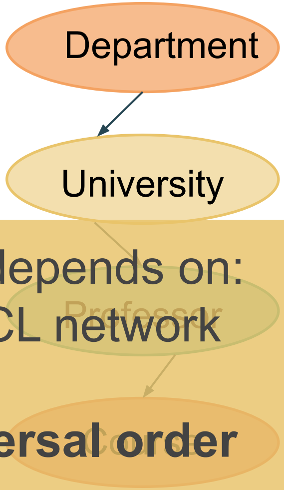
Random Traversal



Data needs to be loaded; Professors validated in the end

Validation Time:
8379 ms

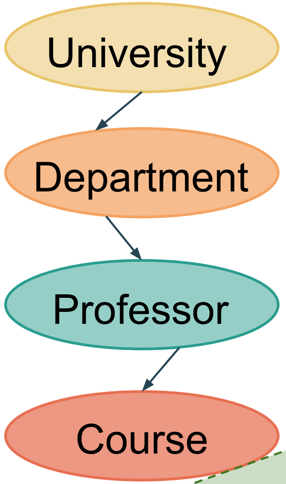
Following Links



Opportunity to save by following links; Profs and Depts validated after the next shape

Validation Time:
6672 ms

Sophisticated Traversal



Use knowledge from previous validations; improved performance

KNOWLEDGE EXPLOITATION

Validation Time:
525 ms

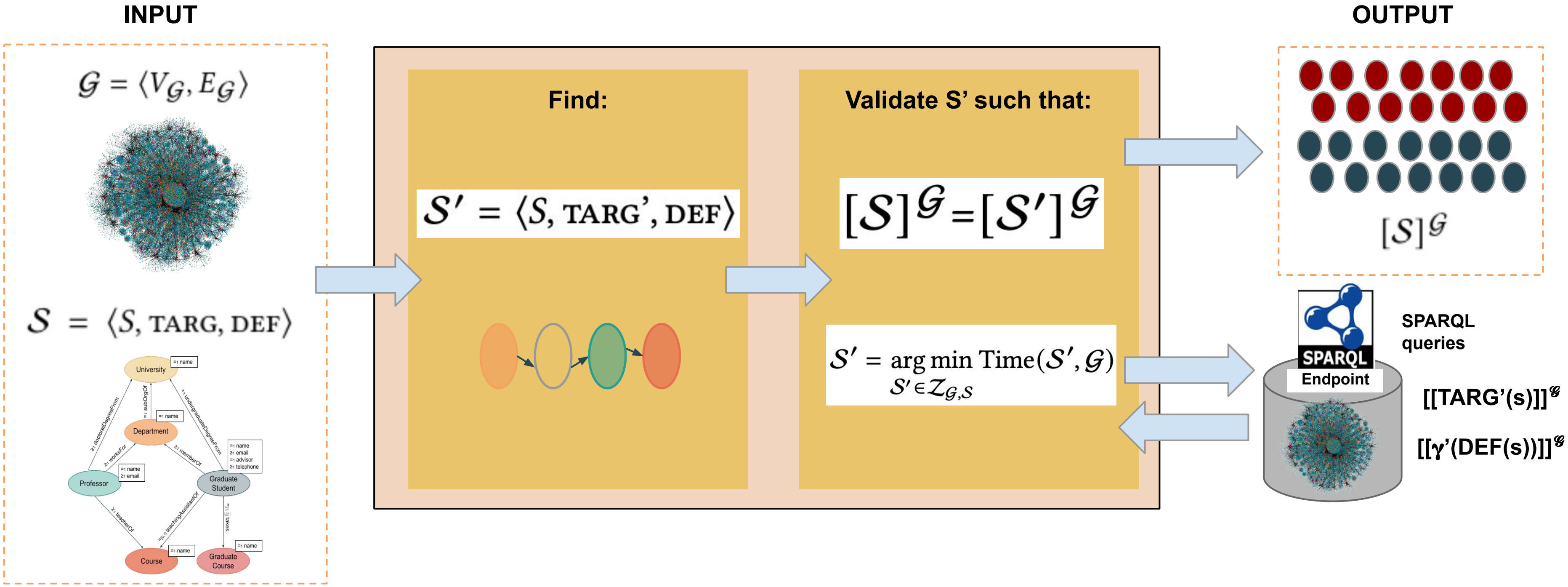
SHACL validation time depends on:

- **size** of KG and SHACL network
- KG **quality**
- SHACL network **traversal order**

Trav-SHACL

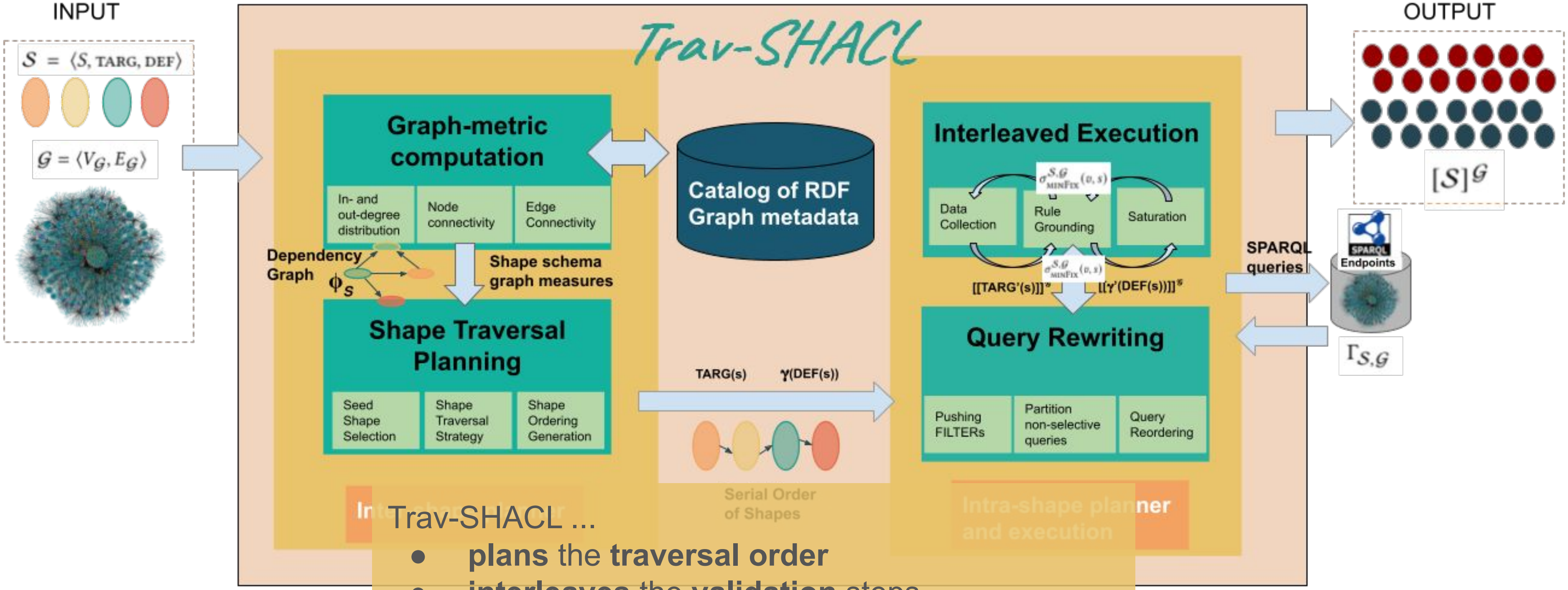
- **SHACL validator over SPARQL endpoints**
- **Assumes RDF graph to validate is free of blank nodes**
- **SHACL shapes are translated into stratified Datalog rules**
- **Efficient validation of knowledge graphs**
    - Interleaved execution
    - Query rewriting
    - Planning of traversal order
- **Continuous generation of results**
- **Only JSON input so far instead of the standard**
- **SHACL fragments** express **recursive** networks **without negation**

Trav-SHACL: Validating Integrity Constraints



Trav-SHACL: Validating Integrity Constraints

https://github.com/SDM-TIB/Trav-SHACL

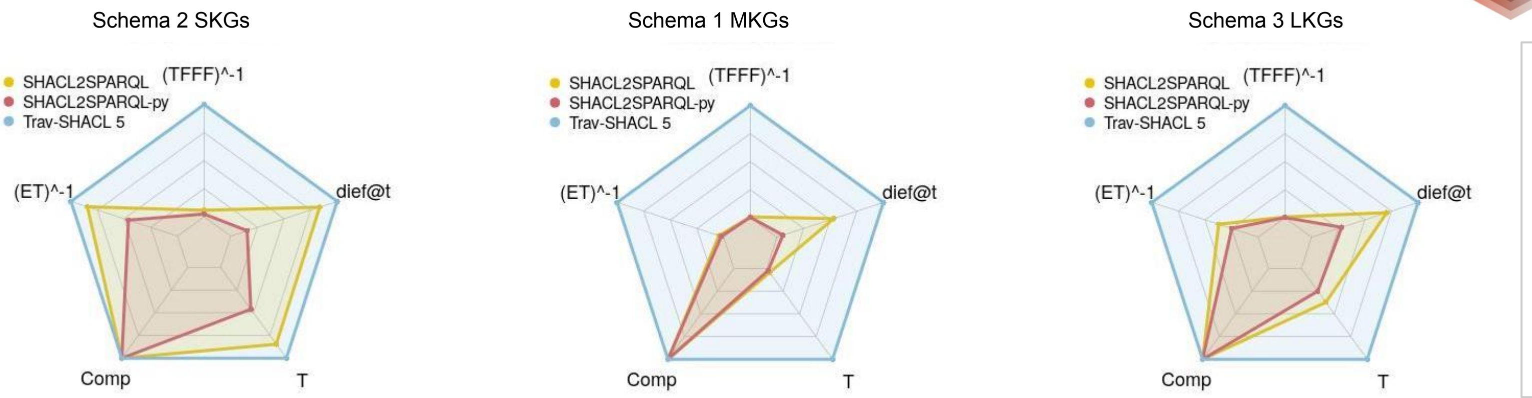


Trav-SHACL ...

- **plans** the **traversal order**
- **interleaves** the **validation** steps
- **rewrites queries** to make them more selective

Monica Figuera, Philipp D. Rohde, and Maria-Esther Vidal. Trav-SHACL: Efficiently Validating Networks of SHACL Constraints, The Web Conference (WWW 2021). https://doi.org/10.1145/3442381.3449877

Trav-SHACL: Experimental Results



Metrics

dief@t: continuous efficiency at time t

(TFFF)^-1: Time for First Answer (sec)

(ET)^-1: Execution Time (sec)

Comp: sum of (in)validated entities

T: Throughput (answer/sec)

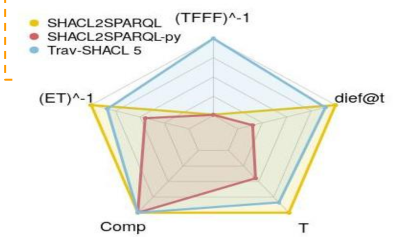
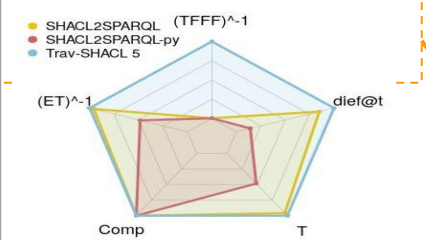
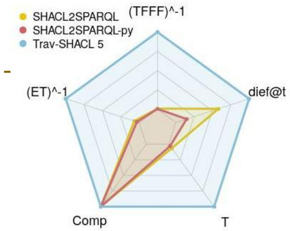
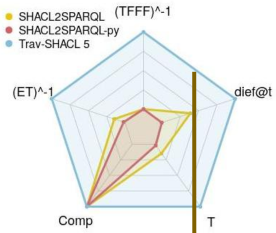
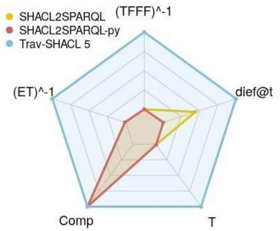
- # Constraint query mappings:
  - **839K** in SHACL2SPARQL,
  - **468K** in Trav-SHACL.
- # Constraint query mappings:
  - **703K** in SHACL2SPARQL,
  - **814** in Trav-SHACL.
- # Constraint query mappings:
  - **22.94M** in SHACL2SPARQL,
  - **6.19M** in Trav-SHACL.

- **Trav-SHACL always delivers results continuously**,
- generates the first answer faster,
- finishes the execution faster,
- scales up to large knowledge graphs.

Impact of the interleaved execution

Mónica Figuera, Philipp D. Rohde, and Maria-Esther Vidal. Trav-SHACL: Efficiently Validating Networks of SHACL Constraints, The Web Conference (WWW 2021). https://doi.org/10.1145/3442381.3449877

Continuous Behavior

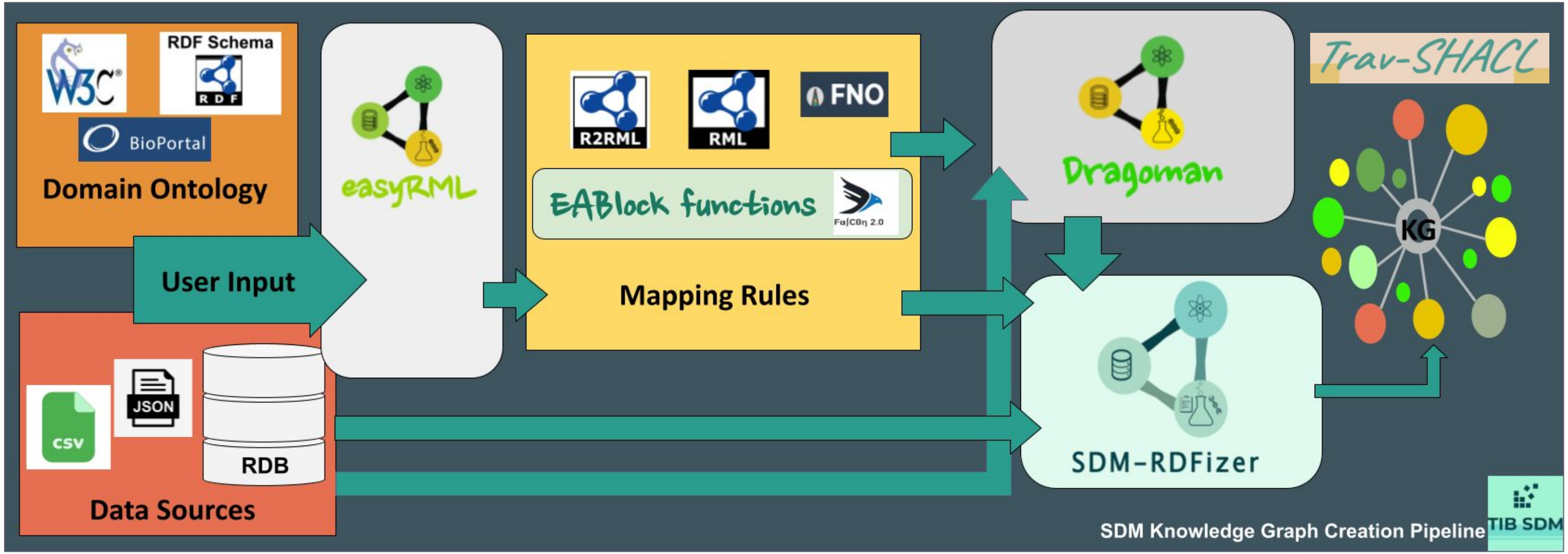
	Small KG	Medium KG	Large KG
High Quality			Steadier Answer Generation
Low Quality			

Prominent Difference

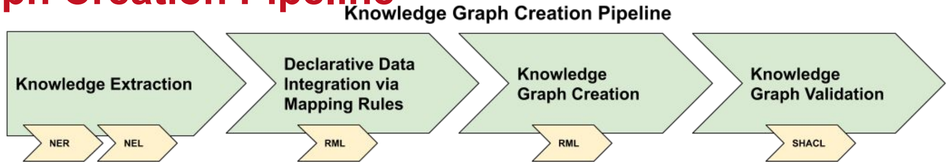
Mónica Figuera, Philipp D. Rohde, and Maria-Esther Vidal. Trav-SHACL: Efficiently Validating Networks of SHACL Constraints, The Web Conference (WWW 2021). https://doi.org/10.1145/3442381.3449877

Pipeline for Knoowledge Graph Creation

The SDM Knowledge Graph Creation Pipeline

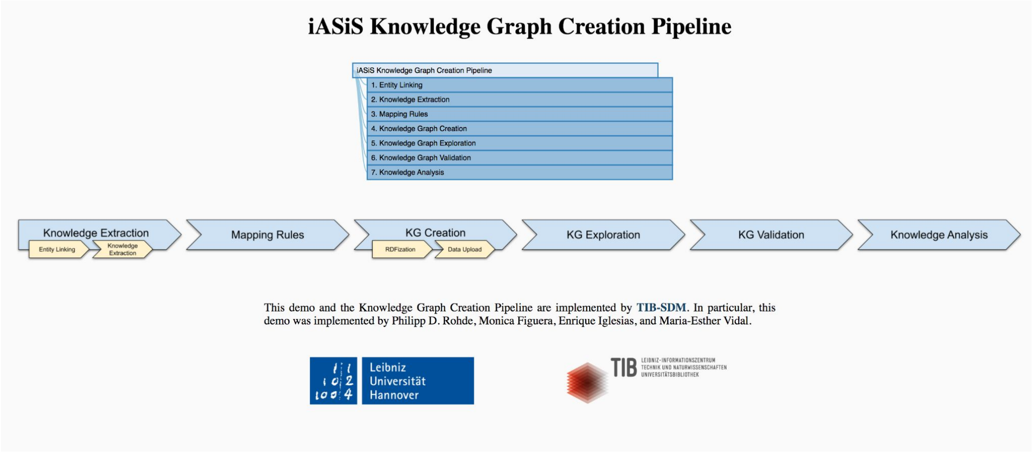


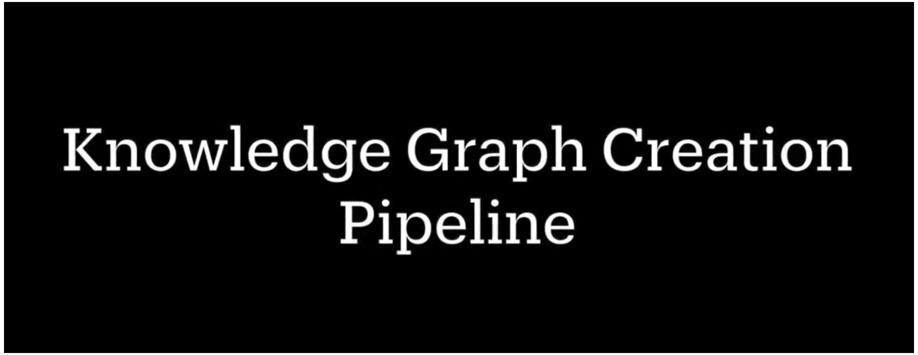
Knowledge Graph Creation Pipeline



Tool	Main Features
FALCON [Sakor et al. 2019,2020]	• Links **surface forms** in a short text into entities in Knowledge Graphs (KGs) • Guided by rules of **English** morphology, and tokenization and compounding Resorts to **alignments** among entities, their labels, and definition in existing KGs (e.g., DBpedia, Wikidata, and UMLS) for disambiguation
SDM-RDFizer [Iglesias et al. 2020,Jozashoori et al. 2020]	• **RML compliant** engine to create KGs • Implements **RML** mapping rules with a set of **non-blocking** operators
EABlock Functions [Jozashoori et al. 2022]	• **Toolbox** of functions for Entity Alignment to be included in **RML mapping rules** • Functions perform named entity recognition over short text and entity linking to **DBpedia, Wikidata,** and **UMLS**
Trav-SHACL [Figuera, Rohde, Vidal 2021]	• W3C recommendation language for specifying integrity constraints over RDF KGs • A **SHACL** engine to validate constraints over KGs • Implements **non-blocking** validation **strategies**

Demo and Video





Prepared by Akhilesh Vyas https://service.tib.eu/ldmservice/dataset/sdmkgc

BioFalcon

- Entity & Relation linking to UMLS
- Hybrid approach
  - Linguistic rules
  - Semantic type prediction model
- Receive input text from the user
- Extract & link the extracted entities to UMLS
- Available as an online API

GitHub https://github.com/SDM-TIB/falcon2.0

Demo: https://labs.tib.eu/sdm/biofalcon/
https://service.tib.eu/ldmservice/service/falcon-demo

Ahmad Sakor, Isaiah Onando Mulang', Kuldeep Singh, Saeedeh Shekarpour, Maria-Esther Vidal, Jens Lehmann, Sören Auer. **Old is Gold: Linguistic Driven Approach for Entity and Relation Linking of Short Text**. NAACL 2019

Ahmad Sakor, Kuldeep Singh, Anery Patel, Maria-Esther Vidal. **Falcon 2.0: An Entity and Relation Linking Tool over Wikidata**. CIKM 2020

BioFalcon

https://labs.tib.eu/sdm/biofalcon/

Type what you wanna BioFalcon to catch

The serum concentration of Lepirudin can be decreased when it is combined with Garlic

The risk or severity of adverse effects can be increased when Lepirudin is combined with Deoxycholic Acid

Forschungszentrum · Research Center L3S

Leibniz Universität Hannover

TIB LEIBNIZ INFORMATION CENTRE FOR SCIENCE AND TECHNOLOGY UNIVERSITY LIBRARY

developed by Ahmad Sakor

SDM Knowledge Graph Creation Pipeline

Samaneh Jozashoori,
Ahmad Sakor,
Enrique Iglesias

easyRML

- Facilitates the RML Mapping rule generation
- Receives mappings data from the user via a user interface and translate it into a validated turtle file including RML mapping rules
- Omits the overhead of syntax verification and errors from the user side

GitHub https://github.com/SDM-TIB/easyRML

Demo: https://tib.eu/cloud/s/rFYL3CZHqYSQjFC

Dragoman

- An Optimized, RML-engine-agnostic Interpreter for Functional Mappings
- Plans the optimized execution of FnO functions integrated in RML mapping rules
- Interprets and transforms mappings into function-free rules that can be translated into RDF using any RML-compliant engine

★ Users can easily add their own scripts defining new functions
★ It can be adopted by any RML-compliant knowledge graph creation pipeline
★ Able to interpret composite functions
★ Able to interpret the list of outputs (which is the limitation of current RML language)
★ Is efficient (using optimization techniques) in terms of execution time

GitHub https://github.com/SDM-TIB/Dragoman

Demo: https://tib.eu/cloud/s/ikjiHyf8RNrEHSY

Samaneh Jozashoori, David Chaves-Fraga, Enrique Iglesias, Maria-Esther Vidal and Oscar Corcho. **FunMap: Efficient Execution of Functional Mappings for Scaled-Up Knowledge Graph Creation**. ISWC 2020

SDM-RDFizer

GitHub https://github.com/SDM-TIB/SDM-RDFizer

Demo: https://www.youtube.com/watch?v=DpH_57M1uOE

Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, Maria-Esther Vidal.
SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs. ACM CIKM 2020.

Hands-on Goals

- Understand the process of knowledge graph creation
- Understand the entity linking and knowledge extraction processes
- Define and execute simple mapping rules
- Define and execute mapping rules with functions such as entity linking functions
- Create a knowledge graph

The SDM Knowledge Graph Creation Pipeline (Docker Install)

For Linux:

> sudo curl -L "https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

> sudo chmod +x /usr/local/bin/docker-compose

> docker-compose --version

More Info:

https://docs.docker.com/compose/install/

For Windows :

https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe

More Info:

https://docs.docker.com/docker-for-windows/install/

For Mac :

https://download.docker.com/mac/stable/Docker.dmg

More Info:

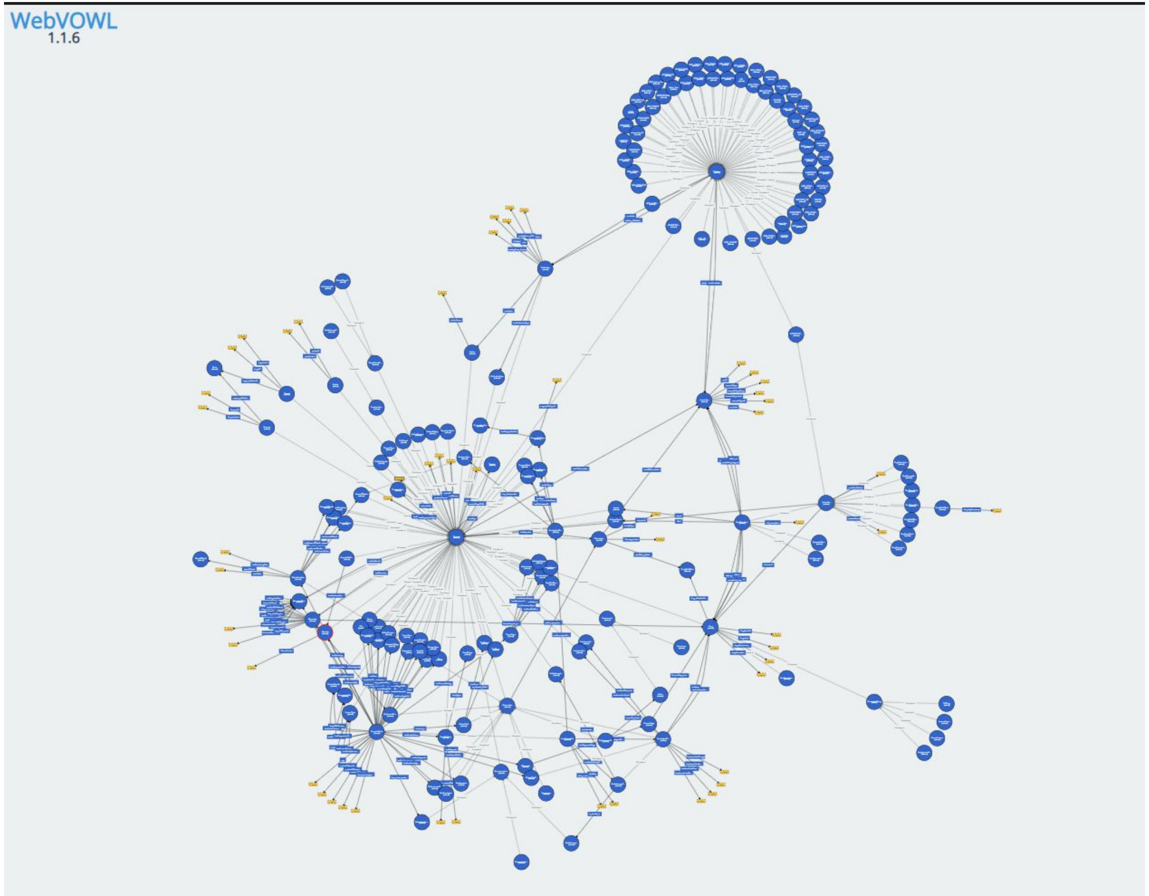
https://docs.docker.com/docker-for-mac/install/

The SDM Knowledge Graph Creation Pipeline

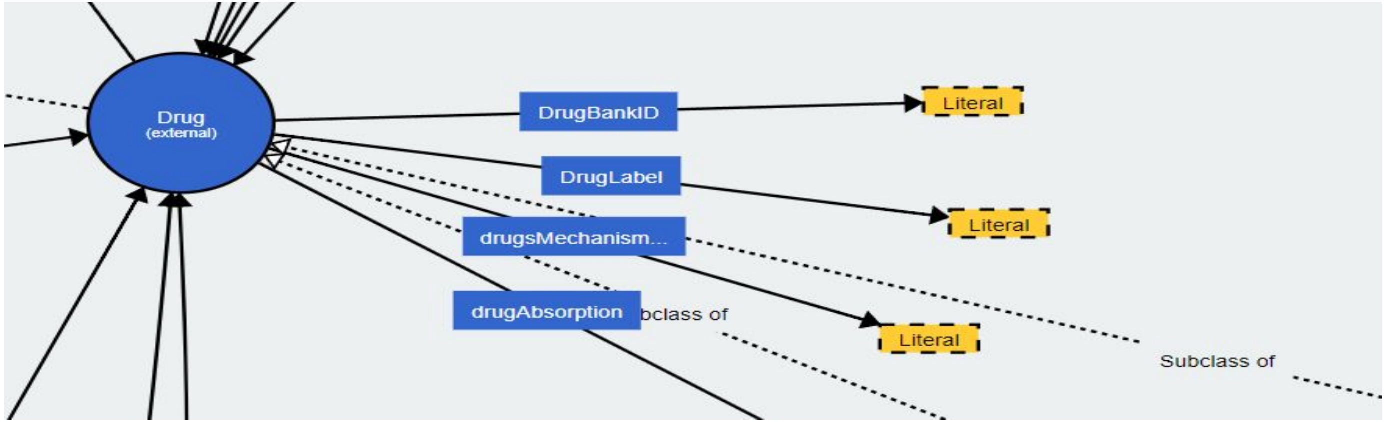
```
> mkdir kgc_2021_tutorial
> cd kgc_2021_tutorial/
> git clone https://github.com/SDM-TIB/KGC_Workshop_2021.git
```

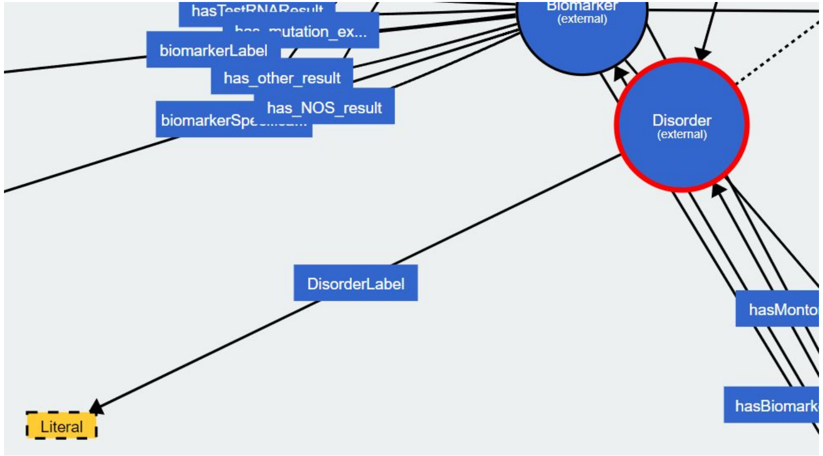
```
> cd KGC_Workshop_2021
> docker network create kgc_2021
> docker-compose up -d
> docker ps
```

Example- A Unified Schema



Example- A Unified Schema





Assignments- Defining New Knowledge Mappings

Task 1: Create knowledge mappings in RML using easyRML

```
@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 xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix base: <http://tib.de/ontario/mapping#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix sdmkgc: <https://github.com/SDM-TIB/KGC/vocab/> .

<drug_TriplesMap1>
  rml:logicalSource [ rml:source "/data/drug.csv";
                      rml:referenceFormulation ql:CSV
                    ];
  rr:subjectMap [
    rr:template "https://github.com/SDM-TIB/KGC/entity/{DrugBankID}";
    rr:class sdmkgc:Drug
  ];
  rr:predicateObjectMap [
    rr:predicate sdmkgc:drugLabel;
    rr:objectMap [
      rml:reference "DrugName";
    ]
  ].
```

Assignments- Defining New Knowledge Mappings

Task 1: Understanding and creating knowledge mappings

I. Open this URL in your browser http://localhost:5000 and create the mapping file

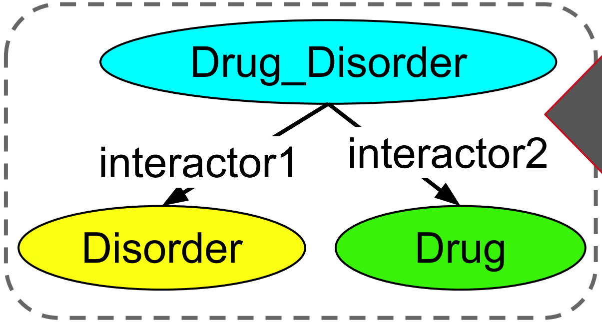
Demo: https://tib.eu/cloud/s/rFYL3CZHqYSQjFC

II. You can use the ontology and data file available at data folder

III. Provide the output path in easyRML interface e.g. `/easyRML/sources/`

IV.
```
> cd /easyrml
> ls -l
> less given_name_to_the_mapping_file.ttl
```

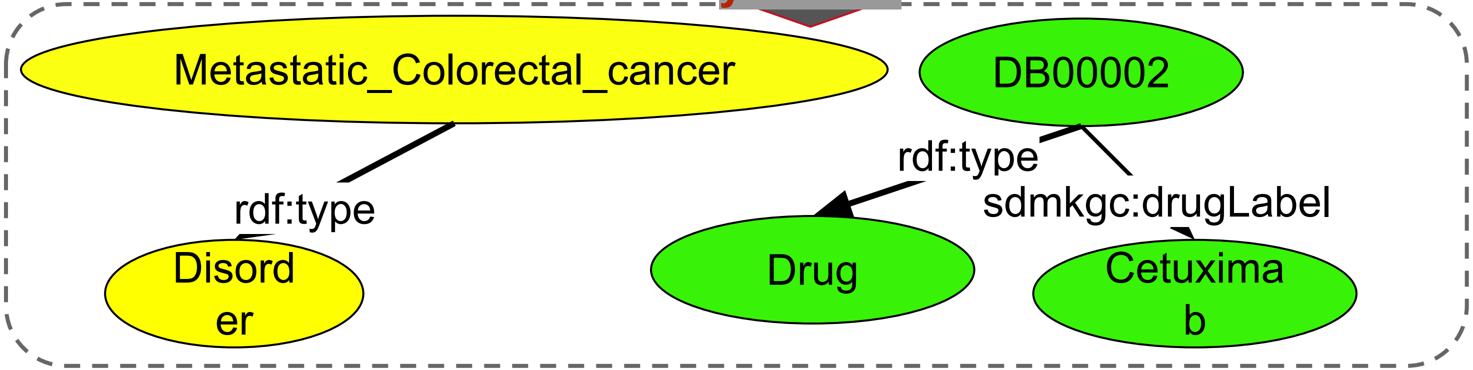
Assignments- Entity Linking Integrated in the Knowledge Mapping



How you want to model this data and integrate into your KG!!

Drug_Disorder_Dataset	
Drug	Disorder
C0995188	C0948380
C0034266	C0376358

What you already have in your KG!



Assignments- Entity Linking as pre-processing

Task 2: Execute an entity linking as pre-processing

a) Analyze Entities to be Linked

```
> cd data
> ls -l
> less drugs.csv
> cd ..
```

```
> docker exec -it kgc_workshop_2021 python3 /tutorial/src/drugs_umls_link.py
```

b) Execute the Falcon entity linking component

c) A

```
> cd data
> ls -l
> less drugs.csv
> cd ..
```

Assignments- Entity Linking Integrated in the Knowledge Mapping

Task 3: Execute knowledge mappings in which the entity linking is integrated as functions

I.

```
> cd data
> ls -l
> less drug_disorder.csv
> cd ..
```

II. Apply Dragoman to execute functions in knowledge mappings and ... e

```
> less configs/config-Dragoman.ini
> docker exec -it kgc_2021_dragoman python3 /app/run_translator.py
/app/configs/config-Dragoman.ini
```

III.

```
> cd dragoman
> ls -l
> less drug_disorder_transferred_mapping.ttl
> cd ../..
```

Assignments- Generate KG based on defined knowledge mappings

Task 4: Execute knowledge mappings in which the entity linking is integrated as functions

I. Apply SDM-RDFizer to execute generate rdf triples (KG) (first analyze the required config file and then execute)

```
> less configs/config-SDM-RDFizer.ini
> docker exec -it kgc_2021_semantic_enrichment python3
/app/rdfizer/run_rdfizer.py /app/configs/config-SDM-RDFizer.ini
```

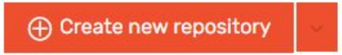
II.

```
> cd rdf/output/rdf
> ls -l
> less drug_disorder.nt
> cd ../..
```

Assignments- Uploading The KG into GraphDB

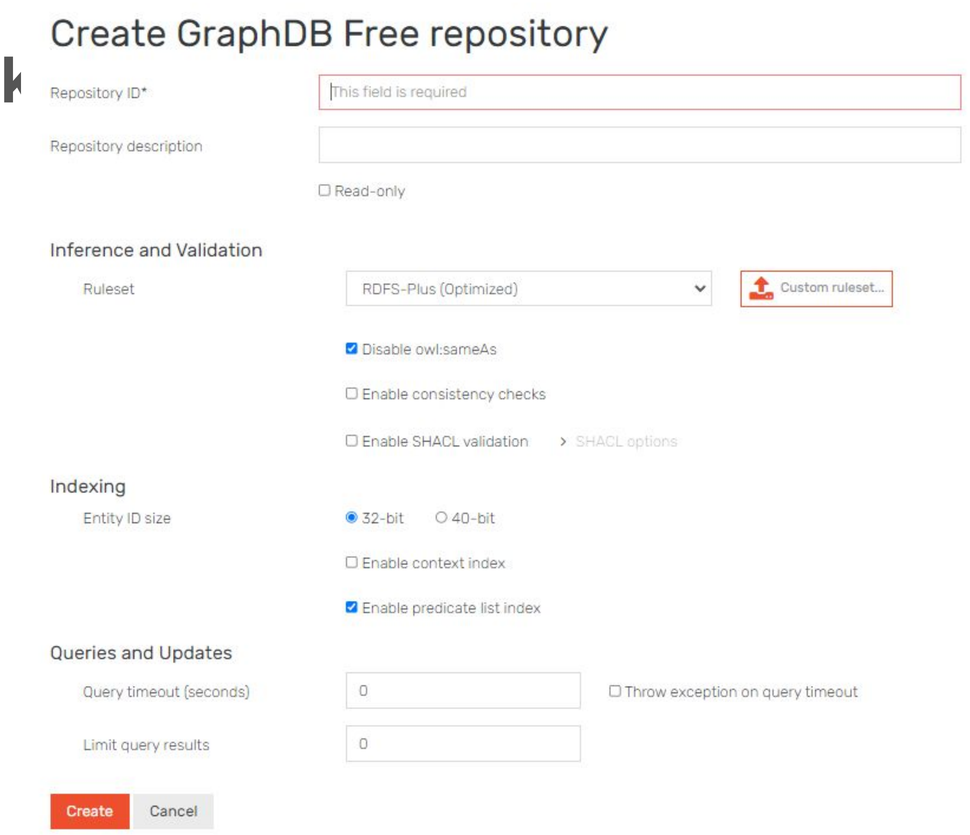
http://localhost:7200/

Task 6: Creating a Knowledge Graph in GraphDB



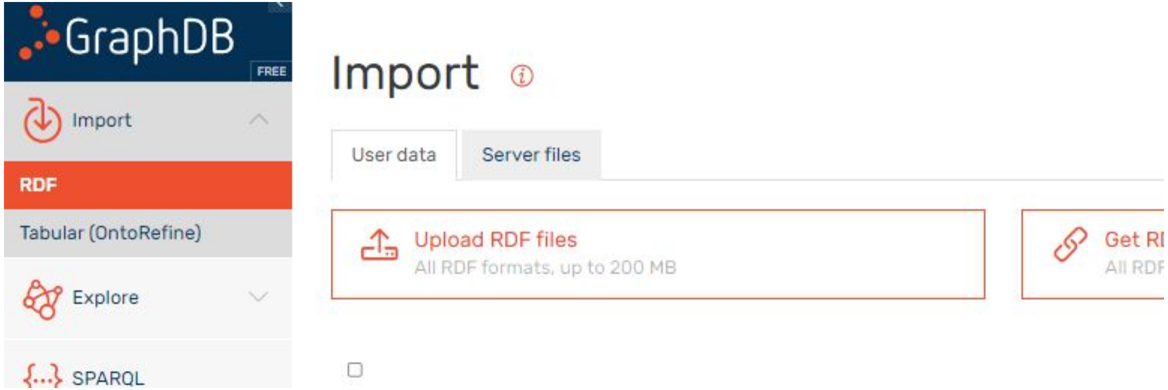
Assignments- Uploading The KG into GraphDB

Task



GraphDB

Task 6: Creating a Knowledge Graph in GraphDB



Validation of Knowledge Graphs

Philipp D. Rohde,
Julian Gercke

Requirements

- Docker
  https://docs.docker.com/get-docker/

- docker-compose
  https://docs.docker.com/compose/install/

Getting Started

1. **Clone the repository**

```
git clone https://github.com/SDM-TIB/KGV_Workshop_2021.git
```

2. **Start the containers**

```
cd KGV_Workshop_2021
docker-compose up -d
```

3. **Wait for the containers to start**

   You can check the status of the endpoint by visiting
   http://localhost:15000/sparql

University Data

- LUBM benchmark
- One University
- 14 Classes, e.g.,
  - Full/Assistant/Associate Professors
  - (Under-)Graduate Students
  - (Graduate) Courses
  - Departments
  - Publications

Task 1: Create Constraints

1. **Go to the directory ‘shapes/lubm’**
2. **Open the file ‘University.json’**
3. **Add the following constraint:**
    a. Universities have at most one name
4. **Open the file ‘FullProfessor.json’**
5. **Correct the target query**
6. **Add the following constraints:**
    a. Full professors are teacher of at least one course (shape Course)
    b. Full professors have exactly one name
    c. Full professors have at least one email address (ub:emailAddress)

Task 1: Create Constraints (Cont.)

7. **Open the file ‘Department.json’**
8. **Add the following constraints**
    a. Departments are a sub-organization of exactly one University (shape University)
    b. Departments have exactly one name
9. **Open the file ‘Publication.json’**
10. **Add the following constraint**
    a. A publication does not have an undergraduate student as author

Task 2: Knowledge Graph Validation

1. **Go to http://localhost:5001/validate**

2. **Enter the required details**
   a. URL of the knowledge graph
   b. Path with the shapes

3. **Validate the knowledge graph**

4. **Examine the result**

Future Directions

Transparency in Knowledge-driven Data Ecosystems-Example

Lung Cancer Protocols:

Afatinib is a second generation Tyrosine Kinase Inhibitors (TKI) **not** recommended for non-small cell lung cancer patients with **Epidermal Growth Factor Receptor (EGFR)** mutation negative.

Lapatinib is a dual Tyrosine Kinase Inhibitors (TKI) for non-small cell lung cancer patients with **HER2** mutation positive or **EGFR** positive.

Instances of a knowledge graph:

```
ex:patient1 rdf:type ex:NSCLG-EGFR-negative .
ex:patient1 rdf:type ex:NSCLG-HER2-OR-EGFR-positive .
ex:patient1 ex:hasOncologicalTreatment dbpedia:Afatinib .
```

Querying Declarative Mapping Rules

```
PREFIX rr:  <http://www.w3.org/ns/r2rml#>
PREFIX rml: <http://semweb.mmlab.be/ns/rml#>
PREFIX ex:  <http://ex.org/vocab/>

SELECT DISTINCT ?class ?mappingRule ?logicalSource ?predicate ?sourceAttribute
WHERE {
?mappingRule rml:logicalSource ?ls.
?ls          rml:source        ?logicalSource.
?mappingRule rr:subjectMap     ?subject.
?subject     rr:class          ?class.
FILTER (?class in (ex:NSCLG-EGFR-negative, ex:NSCLG-HER2-OR-EGFR-positive))
OPTIONAL {   ?mappingRule rr:predicateObjectMap ?pObjectMap .
             ?pObjectMap  rr:predicate          ?predicate .
             ?pObjectMap  rr:objectMap          ?objectMap .
             ?objectMap   ?mode                 ?sourceAttribute}}
```

SPARQL Query to Retrieve RML Mapping Rules defining the classes **`ex:NSCLG-EGFR-negative`** and **`ex:NSCLG-HER2-OR-EGFR-positive`**

Validating Integrity Constraints

Instances of a knowledge graph:

```
ex:patient1 rdf:type ex:NSCLG-EGFR-negative .
ex:patient1 rdf:type ex:NSCLG-HER2-OR-EGFR-positive .
ex:patient1 ex:hasOncologicalTreatment dbpedia:Afatinib .
```

Evaluation of SHACL shapes enable the validation of the protocols

```
ex:NSLCProtocol1
    a sh:NodeShape ;
    sh:targetClass ex:NSLC-EGFR-negative ;
    sh:property [
        sh:path ex:hasOncologicalTreatment ;
        sh:hasValue dbpedia:Afatinib;
        sh:maxCount 0 ]
```

```
ex:NSLCProtocol2
    a sh:NodeShape ;
    sh:targetClass ex:NSLC-HER2-OR-EGFR-positive ;
    sh:property [
        sh:path ex:hasOncologicalTreatment ;
        sh:hasValue dbpedia:Lapatinib;
        sh:minCount 1 ]
```

But…. the evaluation of SHACL shapes (or any other language) does not allow tracing and explaining the invalidation of the constraints.

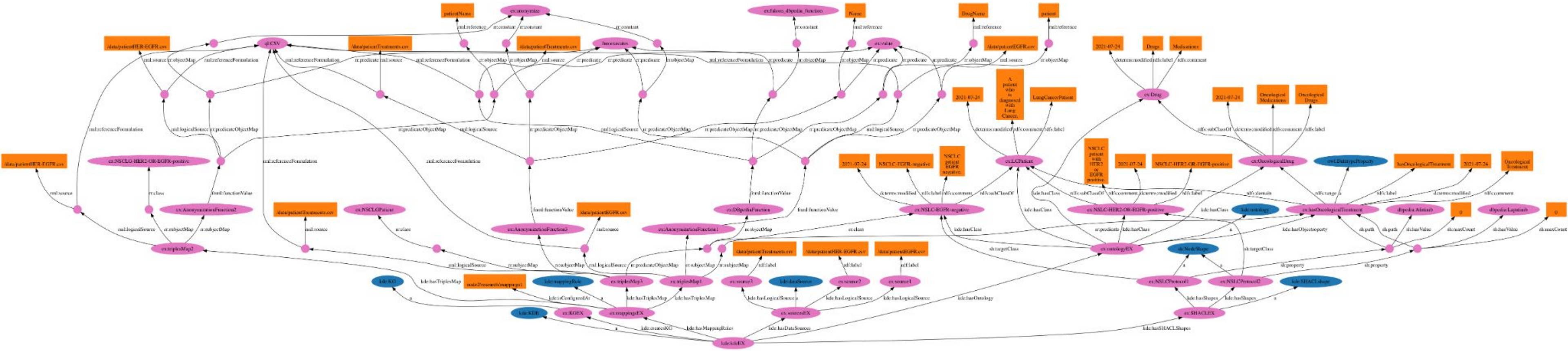
Provenance Meta-data to Enhance Transparency

RDF-star (or Property) graphs to describe how RDF triples have been computed

```
<<subject property object>>
  prov:wasGeneratedBy         triplesMap;
  prov:generaredAtTime        time-stamp;
  kde-prov:subjectRawValue    subjectValue;
  kde-prov:objectRawValue     objectValue.
```

```
<<ex:patient1 rdf:type ex:NSCLG-EGFR-negative>>
  prov:wasGeneratedBy         ex:triplesMap1;
  prov:generaredAtTime        "2021-09-08T08:20:00+06:00"^^xsd:dateTimeStamp;
  kde-prov:subjectRawValue    "John Smith".

<<ex:patient1 rdf:type ex:NSCLG-HER2-OR-EGFR-positive>>
  prov:wasGeneratedBy        ex:triplesMap2;
  prov:generaredAtTime       "2021-09-08T08:20:00+07:00"^^xsd:dateTimeStamp;
  kde-prov:subjectRawValue   "John Smith".

<<ex:patient1 ex:hasOncologicalTreatment dbpedia:Afatinib>>
  prov:wasGeneratedBy        ex:triplesMap3;
  prov:generaredAtTime       "2021-09-08T08:20:00+08:00"^^xsd:dateTimeStamp;
  kde-prov:subjectRawValue   "John Smith";
  kde-prov:objectRawValue     "atinib".
```

Transparency in Knowledge-driven Data Ecosystems



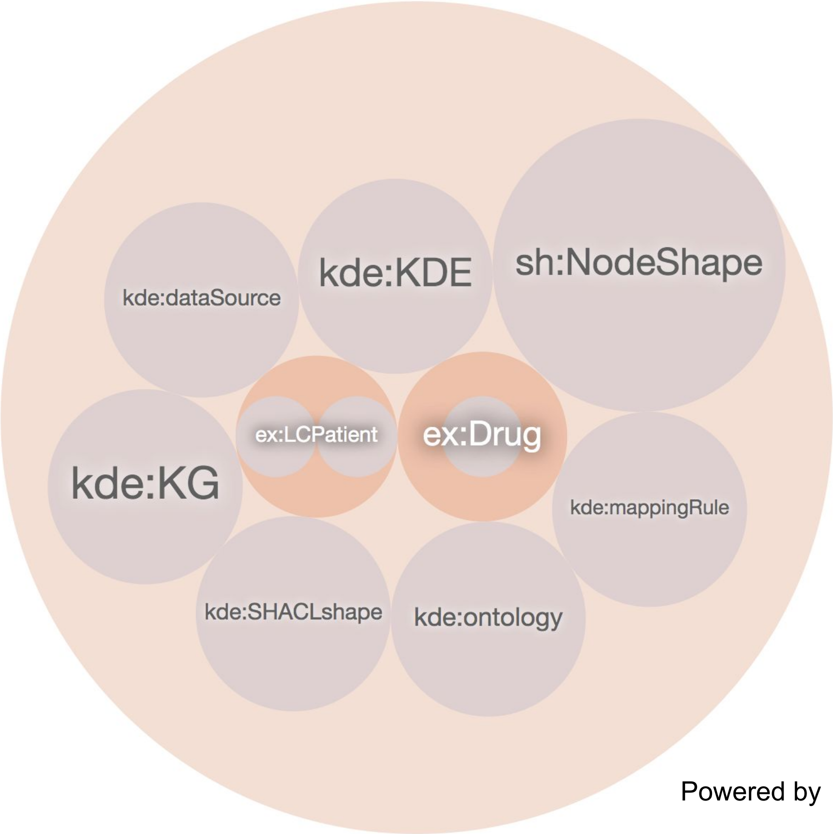
KDE=<O,S,M,IC>

- **O**: Ontology
- **S**: Data Sources
- **M**: RLM + FnO Mappings
- **IC**: SHACL shapes

```
<<ex:patient1 rdf:type ex:NSCLG-EGFR-negative>>
  prov:wasGeneratedBy   ex:triplesMap1;
  prov:generaredAtTime "2021-09-08T08:20:00+06:00"^^xsd:dateTimeStamp;
  kde-prov:subjectRawValue  "John Smith".
<<ex:patient1 rdf:type ex:NSCLG-HER2-OR-EGFR-positive>>
  prov:wasGeneratedBy   ex:triplesMap2;
  prov:generaredAtTime "2021-09-08T08:20:00+07:00"^^xsd:dateTimeStamp;
  kde-prov:subjectRawValue  "John Smith".
<<ex:patient1 ex:hasOncologicalTreatment dbpedia:Afatinib>>
  prov:wasGeneratedBy  ex:triplesMap3;
  prov:generaredAtTime "2021-09-08T08:20:00+08:00"^^xsd:dateTimeStamp;
  kde-prov:subjectRawValue  "John Smith";
  kde-prov:objectRawValue    "atinib".
```

Traceable Knowledge Graph

Classes in Traceable Knowledge Graphs



Powered by GraphDB

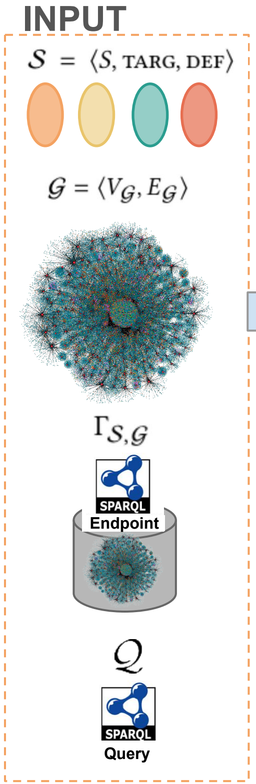
Queries against Traceable Knowledge Graphs

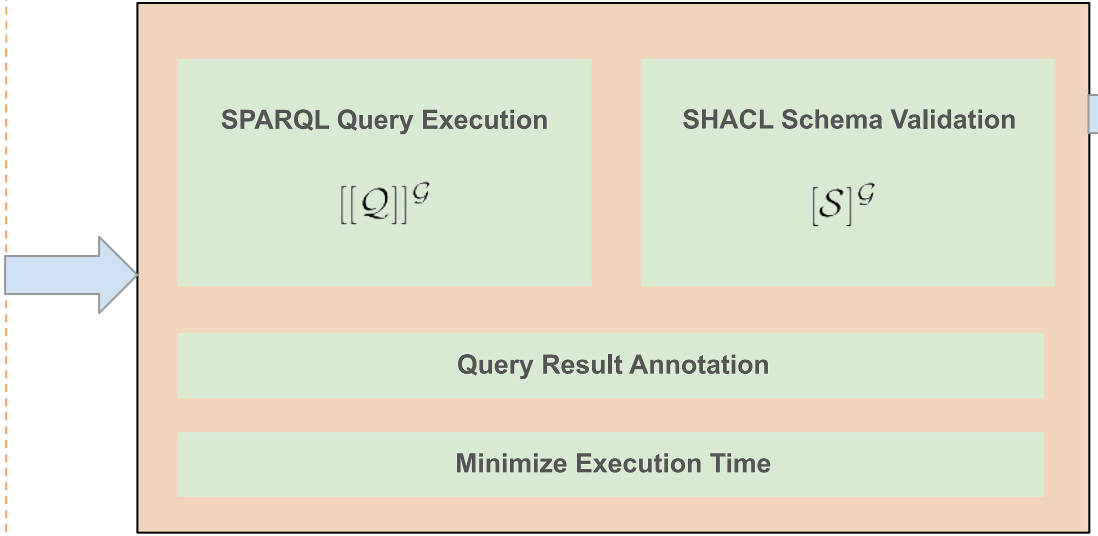
```
PREFIX ex: <http://example.com/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX kde-prov: <http://kde.org/prov#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX rr:  <http://www.w3.org/ns/r2rml#>
PREFIX rml: <http://semweb.mmlab.be/ns/rml#>
SELECT distinct ?p ?tm1 ?subjectValue1 ?objectValue2 ?logicalSource1 ?subject1 ?predicate1 ?objectMap1 ?tm2
WHERE {
  <<?p ex:hasOncologicalTreatment dbpedia:Afatinib>>  prov:wasGeneratedBy       ?tm1;
                                                      kde-prov:subjectRawValue  ?subjectValue1;
                                                      kde-prov:ojectRawValue    ?objectValue2.

  <<?p rdf:type ex:NSCLG-EGFR-negative>> prov:wasGeneratedBy  ?tm2; kde-prov:subjectRawValue ?subjectValue1.

  ?tm1      rml:logicalSource ?logical1.
  ?logical1 rml:source        ?logicalSource1.
  ?tm1      rr:subjectMap     ?subject1.
  OPTIONAL {  ?tm1 rr:predicateObjectMap ?pObjectMap .
              ?pObjectMap  rr:predicate         ?predicate1 .
              ?pObjectMap  rr:objectMap         ?objectMap1 .
              ?objectMap1  ?mode                ?sourceAttribute1} .}
```

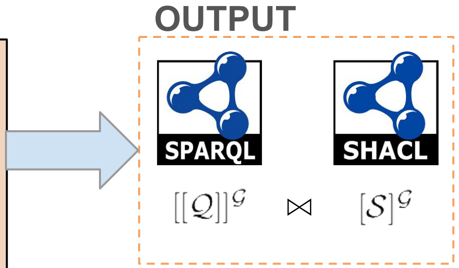
?p	?tm1	?subjectValue1	?objectValue2	?logicalSource1	?subject1	?predicate1	?objectMap1	?tm2
ex:patient1	ex:triplesMap3	John Smith	atinib	/data/patientTreatments.csv	ex:AnonymizationFunction3	ex:hasOncologicalTreatment	ex:DBpediaFunction	ex:triplesMap1

Powered by GraphDB

DeTrusty: Federated Query Engine for Validating KGs







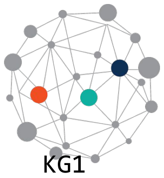
Finding **physical plan** for **query Q** whose execution **validates** the **SHACL shape schema S** and **produces** the **answers of Q efficiently**

Philipp D Rohde SHACL Constraint Validation during SPARQL Query Processing. VLDB Workshop. 2021

Traceable Query Processing

Traditional Approach for Query Processing

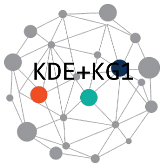
```
Q= SELECT distinct ?p ?drug
WHERE {
  ?p ex:hasOncologicalTreatment ?drug}
```



$[[Q]]^{KG1}$ =
{{(?p,ex:patient1),(?drug,dbpedia:Afatinib)},
{(?p,ex:patient2),(?drug,dbpedia:Vinorelbine)},
{(?p,ex:patient3),(?drug,dbpedia:Nivolumab)}}

Traceable Query Processing

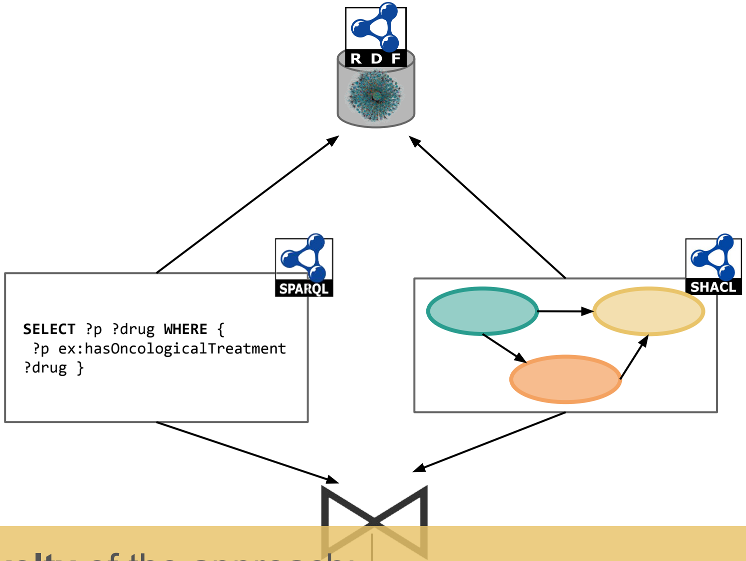
```
Q= SELECT distinct ?p ?drug
WHERE {
  ?p ex:hasOncologicalTreatment ?drug}
```



$[[Q]]^{KDE}_{KG1}$ = {
({(?p,ex:patient1),(?drug,dbpedia:Afatinib)},
{(ex:patient1,dke:invalidates,ex:NSLCProtocol1),
(ex:patient1,dke:invalidates,ex:NSLCProtocol2)},
{{(ex:patient1 prov:wasGeneratedBy ex:triplesMap1),
(ex:patient1 kde-prov:subjectRawValue "John Smith")}
{ (dbpedia:Afatinib prov:wasGeneratedBy ex:triplesMap3),
(dbpedia:Afatinib kde-prov:subjectRawValue "John Smith"),
(dbpedia:Afatinib kde-prov:objectRawValue "atinib")}}),...}

- set of SPARQL mappings
- SHACL validation
- Triple generation explanation

DeTrusty: Federated Query Engine for Validating KGs



Query Decomposition

- subject star-shaped decomposition
- one star ≈ one class

SHACL Validation

- interleaved validation
- subset of shape schema

Query Result Annotation

- add SHACL validation result as metadata
- explainability

Novelty of the approach:

- **identification** of **query plan** able to **combine** query answering with integrity constraint validation
- **explainability** of SPARQL query results
- **optimizations** in SHACL validation

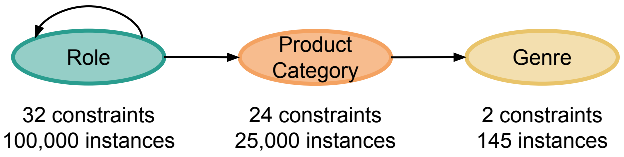
Philipp D Rohde SHACL Constraint Validation during SPARQL Query Processing. VLDB Workshop. 2021

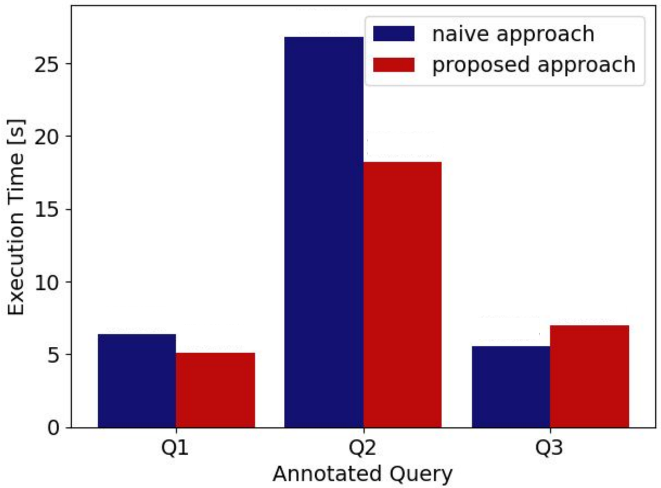
DeTrusty - Initial Results

RQ: Is the performance improved by applying the proposed approach?

WatDiv

- 10 million triples
- 3-5 triple patterns per query
- less than 100 query results

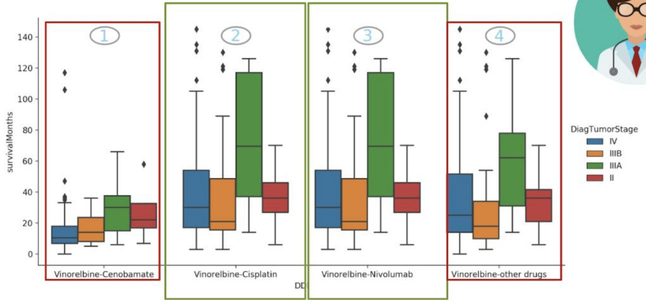




The **performance** is **improved,** but more studies are needed.

Philipp D Rohde SHACL Constraint Validation during SPARQL Query Processing. VLDB Workshop. 2021

What are we still missing?



Case 1: Vinorelbine and **Cenobamate** may interact.

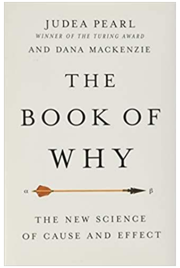
Case 2: Vinorelbine and **Cisplatin** interact, but are there further studies that report the effectiveness of them?

Case 3: Vinorelbine and **Nivolumab** cannot be prescribed together. This must be an error!

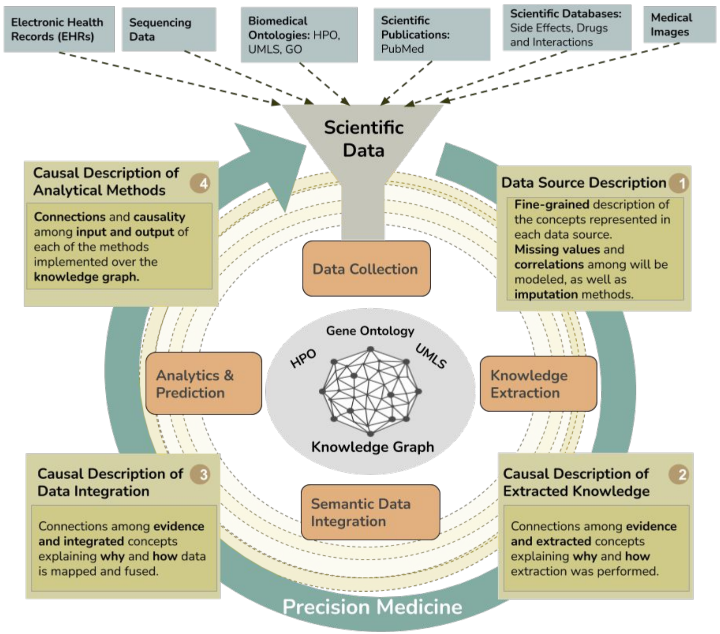
Case 4: Are there further studies that support the effectivenes of **Vinorelbine?**

Causal Queries:

- **Intervention**:
  - What is the effect of the treatment X on Y?
- **Explanation**:
  - Why did Y happen?
- **Counterfactual**:
  - What would happened if instead of X, ¬X?



TRUSTKG



Leibniz Best Minds: Programme for Women Professors

- Data integration paradigm to trace down provenance and causal relations
- Ontologies to document causality and explanation
- Knowledge graph will integrate data, ontologies, and causality models
- Validation and explanation of integrity constraint satisfaction during data collection, curation, integration, and query processing
- Fine-grained representation of scientific publications to support literature-based explanation
- Visualization of explanations of causal relations
- Traceable data privacy regulations
- Evaluated in the context of lung and breast cancer

Lessons Learned

- **Data Integration Systems** state foundations for a declarative specification of KGs
- **Declarative** mapping languages enable data **transparency**, **interpretability**, and **tracking down** data management
- **Planning** the execution of **mapping rules** and **integrity constraints** enable scalable pipelines of KG creation and validation

Follow-up

- **Formalism** to model causality and techniques to mine and explain causal relations on KGs **Fine-grained representation** of (meta)-data
- **Efficient** query processing and management techniques
- **Data Analytical and Machine Learning** methods able to exploit meta-data to enhance explainability

The Knowledge-driven Data Ecosystem as a Resource

Availability

Apache license 2.0 license

Utility

Docker image

Docker Hub

Documentation in

Video

Adoption and Usability

Several European and national funded projects are already using the Knowledge Graph Creation Pipeline

iASiS — Big Data for Precision Medicine

BigMedilytics

CLARIFY

P4-LUCAT

CoyPu

ImProViT

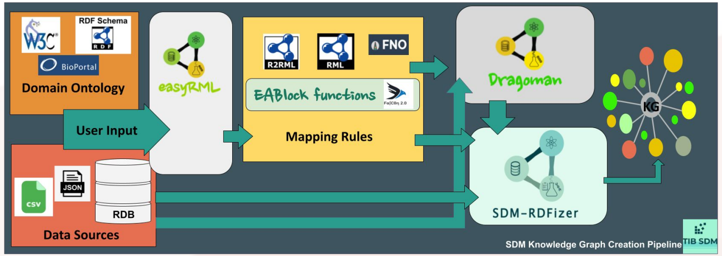
Knowledge 4 COVID-19

PLATOON

https://github.com/SDM-TIB

https://www.youtube.com/watch?v=DpH_57M1uOE

The Scientific Data Management Group



Thanks for your attention!

Maria-Esther Vidal
maria.vidal@tib.eu

@TIB_SDM
@MEVidalSerodio

References

Calvanese et al. Ontop: Answering SPARQL Queries over Relational Databases. Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Semantic Web Journal. 2017 (SWJ 2016 Outstanding Paper Award)

Sequeda and Miranker Ultrawrap: SPARQL execution on relational data. Juan F. Sequeda, Daniel P. Miranker, J. Web Semant. 22: 19-39 (2013)

Priyatna et al. Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. Freddy Priyatna, Óscar Corcho, Juan F. Sequeda. WWW 2014: 479-490

Endris et al. Ontario: Federated Query Processing Against a Semantic Data Lake. Kemele M. Endris, Philipp D. Rohde, Maria-Esther Vidal, Sören Auer. DEXA 2019: 379-395

Chaves et al. Enhancing virtual ontology based access over tabular data with Morph-CSV. David Chaves-Fraga, Edna Ruckhaus, Freddy Priyatna, Maria-Esther Vidal, Óscar Corcho. Semantic Web 12(6): 869-902 (2021)

Iglesias et al. SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs. Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, Maria-Esther Vidal. CIKM 2020: 3039-3046

Jozashoori et al. FunMap: Efficient Execution of Functional Mappings for Knowledge Graph Creation. Samaneh Jozashoori, David Chaves-Fraga, Enrique Iglesias, Maria-Esther Vidal, Óscar Corcho: ISWC 2020: 276-293

Corman et al. 2018. Semantics and Validation of Recursive SHACL. Julien Corman, Juan L. Reutter, Ognjen Savkovic: ISWC 2018: 318-336

References

Dimou et al. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, Rik Van de Walle. LDOW 2014

Arenas-Guerrero et al. Morph-KGC: Scalable Knowledge GraphMaterialization with Mapping Partitions. Julián Arenas-Guerreroa, David Chaves-Fragaa, Jhon Toledoa, María S. Pérezaand Oscar Corcho. Under Evaluation. http://www.semantic-web-journal.net/system/files/swj2924.pdf

Simsek et al. RocketRML - A NodeJS Implementation of a Use Case Specific RML Mapper. Umutcan Simsek, Elias Kärle, Dieter Fensel:. KGB@ESWC 2019: 46-53

Lefrançois et al. Maxime Lefrançois, Antoine Zimmermann, Noorani Bakerally: A SPARQL Extension for Generating RDF from Heterogeneous Formats. ESWC (1) 2017: 35-

Scrocca et al. Turning Transport Data to Comply with EU Standards While Enabling a Multimodal Transport Knowledge Graph. Mario Scrocca, Marco Comerio, Alessio Carenini, Irene Celino. ISWC (2) 2020: 411-429

Geisler S., Vidal M-E, et al. Knowledge-driven Data Ecosystems Towards Transparency. ACM Journal Data and Information Quality. 2022

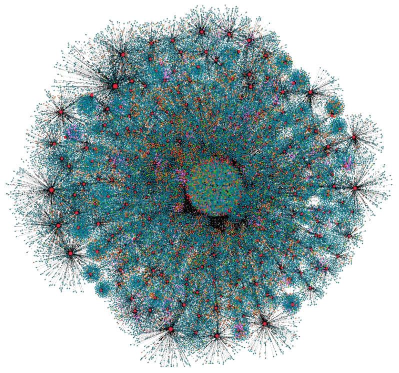
Figuera et al.   Trav-SHACL: Efficiently Validating Networks of SHACL Constraints. Mónica Figuera, Philipp D. Rohde, Maria-Esther Vidal. WWW 2021: 3337-3348

Rohde. SHACL Constraint Validation during SPARQL Query Processing. Philipp D. Rohde. PhD@VLDB 2021

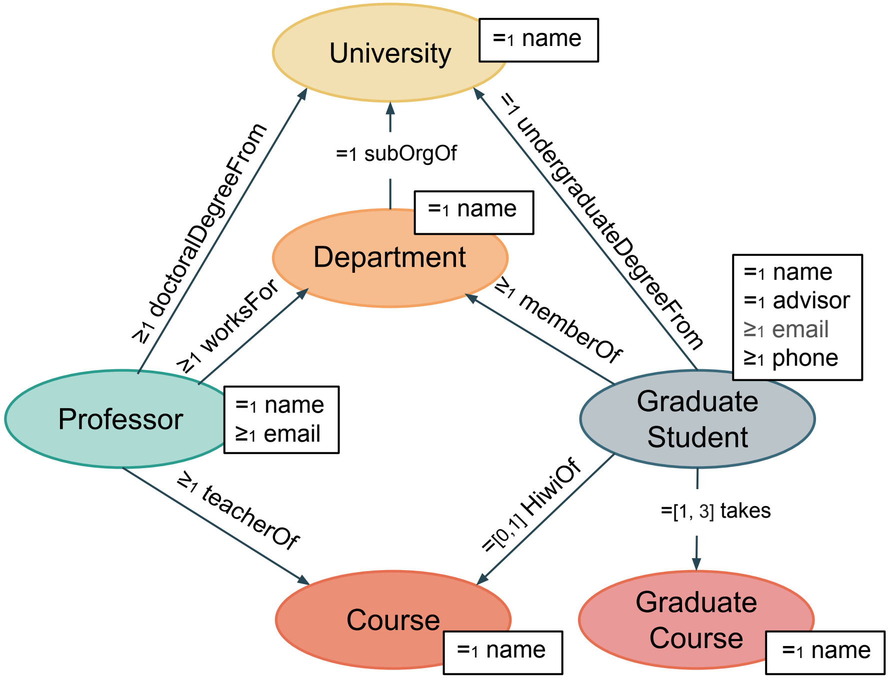
Example

INPUT

KG of a University System
with 37,419 entities
(~1M triples)

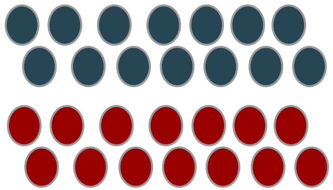


Shapes Schema:
Integrity Constraints on the KG



OUTPUT

Set of classified
(valid / invalid) entities



Trav-SHACL: Reordering of a Shape’s Integrity Constraints

Min-constraints SPARQL query:

```
SELECT DISTINCT ?x, ?uni WHERE {
    ?x ub:subOrgOf ?uni.
    { SELECT DISTINCT ?x WHERE {
        ?x ub:name ?name.
    }}
}
```

Max-constraint SPARQL query:

```
SELECT DISTINCT ?x, ?uni_1, ?uni_2 WHERE {
    ?x ub:subOrgOf ?uni_1.
    ?x ub:subOrgOf ?uni_2.
    FILTER(?uni_1 != ?u
    { SELECT
        ?x ub:
    }}
}
```

EXCLUDED FROM VALIDATION

1. Define and evaluate single min-constraint query

containing all local and inter-shape min-cardinality constraints of shape s

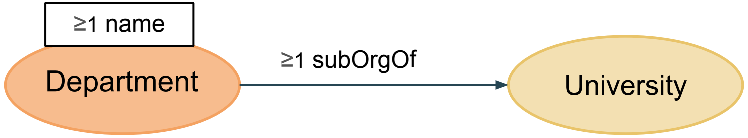
2. Define and evaluate max-constraint queries

negating each max-cardinality restriction

3. Skip needless max-constraint queries

by determining inter- max-constraint violations from constraint’s lower bound

Trav-SHACL: Query Rewriting




```
SELECT DISTINCT ?x, ?uni WHERE {
  ?x rdf:type ub:Department.
  { SELECT DISTINCT ?x, ?uni WHERE {
      VALUES ?uni {ub:LUH}.
      ?x ub:subOrgOf ?uni.
      { SELECT DISTINCT ?x WHERE {
        ?x ub:name ?name.
      }}
  }}
} ORDER BY ?x LIMIT 10000 OFFSET 0
```

Min-constraints SPARQL query γ(DEF(Department))

Target-based Query Rewriting

adds selectivity to γ(DEF(s))

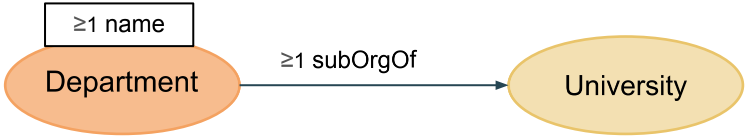
Pushing FILTERs

filters TARG(s) and γ(DEF(s)) with knowledge from out-neighbor shape

Partition of Non-selective Queries

uses SPARQL `LIMIT` and `OFFSET` clauses

Trav-SHACL: Interleaved Execution




```
University(X).
Department(X) <- University(Y), hasDep(Y,X).
```

```
University(“LUH”).
Department(“Informatics”) <- University(“LUH”),
                             hasDep(“LUH”,Informatics”).
```

“LUH” entity is valid, and
hasDep(“LUH”,Informatics”)

Department(“Informatics”) is inferred and
“Informatics” is valid.

Collect Data

from evaluation of TARG(s) and γ(DEF(s))

Ground Logic Rules

for entities and relations between neighboring shapes

Saturate

to classify every v in $\mathcal{V}_{\mathcal{G}}$ until $\sigma^{\mathcal{S},\mathcal{G}}_{\text{MINFIX}}(v, s)$

Trav-SHACL: Execution Time

Shapes Schema Traversal Identified by Trav-SHACL:

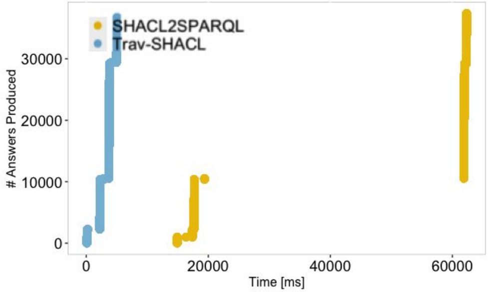
Trav-SHACL validation:

- sound validation result,
- less memory consumption,
- faster execution time,
- scalable to large KGs

Validation Summary:

Engine	# Target queries executed	# Constraint queries executed	# Grounded rules	Classified entities	Validation time
SHACL2SPARQL (baseline)	6 target queries	16 non-selective constraint queries	655,788	16,442 / 37,419 valid, 20,977 / 37,419 invalid	62.55 seconds
Trav-SHACL	**9 target queries (3 queries filter out invalid entities)**	**14 selective constraint queries**	**33,210**	**16,442 / 37,419 valid, 20,977 / 37,419 invalid**	**2.61 seconds**
Trav-SHACL summary	Selective queries evaluated		19.75 less grounded rules	Same validation result	23.97 times faster

Answer Traces:



Trav-SHACL performs a non-blocking execution and produces results faster